Эффективный поиск ограниченно-субоптимальных решений задачи многоагентного планирования

А. А. Андрейчук

Российский университет дружбы народов, г. Москва, Россия

Аннотация. В работе рассматривается задача планирования совокупности неконфликтных траекторий для множества агентов, обладающих возможностью совершения действий произвольной продолжительности. Для ее решения предлагаются две ограниченно-субоптимальные модификации алгоритма конфликтно-ориентированного поиска. Результаты проведенных модельных экспериментальных исследований продемонстрировали высокую вычислительную эффективность предложенных модификаций.

Ключевые слова: планирование траектории, граф, многоагентные системы, эвристический поиск, конфликтно-ориентированный поиск.

DOI 10.14357/20718594220106

Введение

Современные автономные многоагентные системы применяются в различных прикладных областях, включая автоматизированные склады [1], транспортные системы [2], сельское хозяйство [3], при ликвидации ЧП [4]. В случае, когда множество автономных агентов оперируют в общем рабочем пространстве, возникает необходимость решения задачи планирования совокупности траекторий, причем таких, что их исполнение не приводит к столкновениям между агентами или с препятствиями.

Несмотря на то, что в реальных практических задачах алгоритмы многогаентного планирования должны учитывать различные аспекты прикладной области, в которой они применяются, они имеют общий принцип работы. Классическим подходом является использование эвристических алгоритмов планирования траектории на графе особой структуры. В качестве вершин графа выступают возможные положения агентов в пространстве, а ребра соответствуют элементарным переходам, вдоль которых агенты могут перемещаться. При планировании совокупности траекторий для множества агентов необходимо учитывать временную компоненту, так как между агентами могут происходить конфликты. В простейшем случае – это одновременное нахождение двух агентов в одной и той же вершине графа, либо одновременный переход по одному и тому же ребру графа.

Необходимость учитывать временную компоненту существенно повышает вычислительную сложность задачи планирования, так как одному положению агента может соответствовать множество состояний в пространстве поиска, которые отличаются лишь моментом времени. В связи с этим большинство существующих на сегодняшний день алгоритмов используют ряд допущений, позволяющих осуществлять планирование в дискретном времени. Считается, что

✉ *Андрейчук Антон Андреевич. E-mail: andreychuk@mail.com*

все действия агентов имеют одинаковую продолжительность и за один такт времени каждый агент может совершить одно действие – либо переход в смежную вершину графа, либо действие ожидания в текущей вершине.

Одним из простейших и наиболее эффективных с вычислительной точки зрения является приоритизированный подход [5]. Основной принцип его заключается в том, что каждому агенту назначается приоритет, и траектории агентов планируются последовательно в порядке приоритета. Для того чтобы между агентами не происходили конфликты, каждый последующий должен избегать агентов с более высоким приоритетом, фактически воспринимая их как динамические препятствия. Высокая вычислительная эффективность сделала приоритизированный подход довольно распространенным решением, зачастую использующимся на практике. Недостатком этого подхода является отсутствие свойств оптимальности и полноты, так как между агентами, фактически, нет кооперативного взаимодействия.

Однако существуют и другие подходы, в том числе те, что обладают этими свойствами. Одним из первых алгоритмов, решающих задачу многоагентного планирования и гарантирующих нахождение оптимальных решений, был алгоритм OD+ID [6]. В своей основе он использует алгоритм A*[7], а также процедуры декомпозиции операторов (англ. Operator decomposition) и детекции независимых агентов (англ. Independence detection). В последующем появились и другие алгоритмы, способные находить оптимальные решения задачи многоагентного планирования, например, CBS [8], M* [9], ICTS [10] и др.

Наибольшее развитие в последние годы получил подход конфликтно-ориентированного поиска, используемый алгоритмом CBS. Для него было разработано большое количество различных модификаций, которые позволяют повысить вычислительную эффективность алгоритма – ICBS [11], CBS-H [12] и др. В работах [13, 14] были предложены модификации алгоритма CBS, ориентированные на поиск субоптимальных и ограниченно-субоптимальных решений. В отличие от субоптимальных решений, которые могут иметь произвольную стоимость, ограниченно-субоптимальные решения характеризуются тем, что их стоимость гарантировано не превышает стоимость оптимального решения более чем в w раз, где w – коэффициент субоптимальности, заранее задаваемый пользователем.

Существенным недостатком всех вышеперечисленных алгоритмов является их упрощенная модель движения агентов. Ограниченный набор возможных действий агентов, вызванный дискретностью времени, приводит к снижению качества (увеличению стоимости) отыскиваемых решений. Этот недостаток был устранен в алгоритмах CCBS [15] и E-ICTS [16].

Наибольший интерес представляет алгоритм CCBS, который позволяет агентам совершать действия произвольной продолжительности. Как и оригинальный алгоритм CBS, алгоритм CCBS обладает свойствами полноты и оптимальности, но при этом имеет довольно низкую вычислительную эффективность в сравнении со многими другими алгоритмами. В первую очередь это связано с более сложной постановкой задачи и необходимостью рассмотрения большого количества альтернативных вариантов для гарантии того, что найденное алгоритмом решение является оптимальным.

Одним из возможных способов устранения проблемы низкой вычислительной эффективности является поиск ограниченно-субоптимальных решений. Разработке такой модификации алгоритма и посвящена данная работа. В ней будут описаны две модификации алгоритма CCBS, которые позволяют находить ограниченно-субоптимальные решения. Также будут проведены модельные экспериментальные исследования, позволяющие сравнить эти модификации друг с другом и оценить прирост вычислительной эффективности в сравнении с оригинальным алгоритмом CCBS.

1. Постановка задачи

В качестве модели пространства поиска используется граф G = <V, E>, где V – множество вершин графа, каждая из которых соответствует некоторой проходимой области рабочего пространства, а E – множество ребер графа, соответствующее элементарным переходам между центрами двух вершин графа. Область, в которой агенты могут оперировать, представляет собой двумерную плоскость, поэтому каждая вершина графа имеет произвольные координаты (x, y), принадлежащие этой плоскости.

Помимо графа, дано множество N агентов $A = \{agent_1, \dots, agent_N\}$. Каждый $agent_i$ обладает собственным стартовым и целевым положением $start_i \in V, goal_i \in V$, $start_i \neq start_j, goal_i \neq goal_j \; \forall i,j \in [1,N]$. Без ограничения общности будем считать, что каждый агент представляет собой диск радиусом r. Каждый агент обладает возможностью перемещения вдоль множества ребер E с постоянной скоростью c. Помимо этого, каждый агент может осуществлять ожидание произвольной продолжительности, находясь в центре одной из вершин множества V. При этом сделаны допущения, что агенты могут мгновенно менять ориентацию, т.е. направление движения, мгновенно ускоряться до скорости c и мгновенно останавливаться. На Рис. 1. изображен пример графического представления задачи многоагентного планирования, где задан граф и 3 агента. Их стартовые положения – A, E, G, целевые – I, J, D соответственно. На осях отмечены координаты вершин.

Имея заданное множество агентов A и граф G, необходимо построить совокупность неконфликтных траекторий $\Pi = \{\pi_1, \dots, \pi_N\}$, в которой каждая траектория π_i позволяет за конечное время перевести $agent_i$ из его стартового положения $start_i$ в целевое $goal_i$. Траектория π_i представляет собой конечную последовательность действий $\pi_i = \{a_1, \dots, a_k\}$. Каждое действие a_j описывается в виде четверки $\{v^j_{begin}, v^j_{end}, t^j_{begin}, t^j_{end}\}$, где v^j_{begin}, v^j_{end} – начальная и конечная вершины, принадлежащие множеству вершин V, а t^j_{begin}, t^j_{end} – моменты начала и окончания совершения действия a_j. При этом для любого пути π_i выполняются следующие условия: $v^1_{begin} = start_i, v^k_{end} = goal_i, v^j_{end} = v^{j+1}_{begin} \; \forall j \in [1, k-1]$. В случае, когда $v^j_{begin} = v^j_{end}$, действие a_j является действием ожидания.

Траектории $\pi_{пи}$ и π_m являются неконфликтными, если не существует ни одного момента времени, в котором расстояние между центрами агентов $agent_n$ и $agent_m$ меньше чем сумма их радиусов, т.е. $\nexists t: dist\big(pos(agent_n, t), pos(agent_m, t)\big) < 2r$, где $pos(agent_i, t)$ – положение центра агента $agent_i$ в момент времени t, а $dist$ – функция

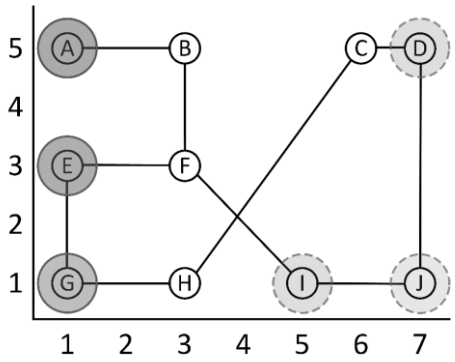

Рис. 1. Пример задачи многоагентного планирования

расчета расстояния между положениями. Аналогичным образом определяется неконфликтность пары действий.

Стоимость траектории π_i соответствует суммарной продолжительности всех действий, которые ее составляют: $cost(\pi_i) = \sum_{j=1}^{k} dur(a_j)$, где $dur(a_j)$ – продолжительность действия a_j, определенная как $dur(a_j) = t^j_{end} - t^j_{begin}$. Критерием оценки качества найденного решения является суммарная стоимость всех траекторий: $cost(solution) = \sum_{i=1}^{N} cost(\pi_i)$. Решение считается оптимальным, если при использовании того же набора входных данных и при тех же допущениях не существует решения меньшей стоимости.

Рассматриваемая задача заключается в следующем. Пусть задан граф *G* и множество *A*, состоящее из *N* агентов. Необходимо построить совокупность неконфликтных траекторий, т.е. таких, которые позволяют всем агентам достичь их целевых положений из их стартовых положений, не создавая конфликтов. При этом стоимость найденного решения должна не превышать стоимость оптимального решения более чем w раз, т.е. должна быть ограниченно-субоптимальной.

2. Ограниченно-субоптимальный алгоритм конфликтно-ориентированного поиска в непрерывном времени

Для решения поставленной задачи предлагается взять за основу алгоритм Continuous-time Conflict Based Search (CCBS) [17], который яв-

ляется модификацией алгоритма конфликтно-ориентированного поиска [8]. Однако, в отличие от оригинального алгоритма, CCBS может решать задачу многоагентного планирования в постановке, когда действия агентов имеют произвольную продолжительность.

Подход алгоритма конфликтно-ориентированного поиска является двухуровневым. На верхнем уровне он оперирует так называемым деревом альтернативных решений. Корень этого дерева – начальное решение, содержащее совокупность траекторий всех агентов, которые были спланированы независимо. На каждом шаге алгоритм извлекает из этого дерева альтернативное решение минимальной стоимости и проверяет его на наличие конфликтов.

Если конфликтов в выбранном альтернативном решении нет, значит искомое решение найдено. Если же в нем содержится хотя бы один конфликт, то создается два новых альтернативных решения, где найденный конфликт устраняется путем наложения ограничения на одного из двух агентов. После того как на агента было наложено ограничение, его траектория перепланируется. Если траектория агента найдена, то получившееся новое альтернативное решение добавляется в дерево.

На каждом шаге алгоритм выбирает текущее лучшее альтернативное решение и осуществляет поиск конфликтов. В случае обнаружения конфликтов создаются два новых альтернативных решения, устраняющие один выбранный конфликт. Процесс продолжается до тех пор, пока не будет найдено решение, не содержащее конфликтов. Благодаря тому, что алгоритм в процессе своей работы рассматривает все доступные альтернативные варианты и делает это в порядке возрастания их стоимости, он обладает свойствами полноты и оптимальности. Доказательство этих свойств смотри в работе [8].

На Рис. 2 показаны примеры конфликтов, возникающие в рассматриваемой постановке задачи. В первом случае конфликт возникает из-за того, что агенты движутся друг за другом, но их направления различаются. В результате чего агент, движущийся позади, «догоняет» переднего агента и происходит конфликт. Во втором случае конфликт возникает из-за того, что агенты движутся по пересекающимся ребрам. В последнем случае, несмотря на то что ребра, по которым движутся агенты, не пересекаются, конфликт возникает из-за их слишком близкого расположения. Помимо представленных типов конфликтов, возможны и стандартные варианты, возникающие в классической постановке задачи многоагентного планирования, когда два агента одновременно движутся по одному и тому же ребру графа, либо находятся в одной и той же вершине графа. Для описания всех возможных типов конфликтов используется набор $<a_i, t_i, a_j, t_j>$, где a_i, a_j – действия агентов i и j, t_i, t_j – моменты начала совершения этих действий.

Как уже упоминалось ранее, для устранения конфликта необходимо наложить ограничение на одного из агентов. При этом недостаточно наложить ограничение лишь на один момент времени, так как если, например, запретить агенту i совершать действие a_i в момент времени t_i, то он сможет совершить его в момент времени $t_i+\varepsilon$, что снова приведет к конфликту. Требуется наложить ограничение на интервал времени, в течение которого агент не может совершать действие. То есть ограничение представляется в виде набора $<i, a_i, [t_{begin}, t_{end})>$, которое запрещает агенту i совершать действие a_i в течение интервала $[t_{begin}, t_{end})$. Для идентификации конфликтов и расчета интервалов ограничений применяется подход, описанный в работе [18]. Он позволяет вычислить минимальное расстояние между парой агентов, имея заданные начальные положения и направления движений.

В качестве планировщика нижнего уровня, т.е. алгоритма, осуществляющего планирование

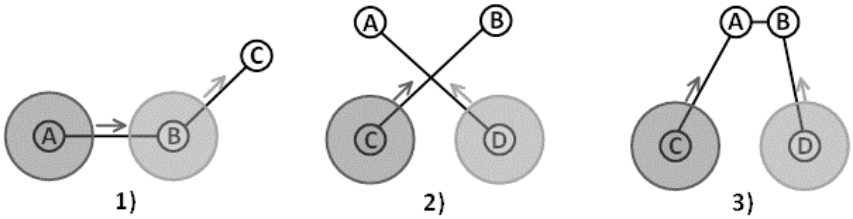

Рис. 2. Примеры конфликтов, возникающие в рассматриваемой постановке задачи

индивидуальных траекторий, используется алгоритм безопасно-интервального планирования (от англ. Safe Interval Path Planning или SIPP) [19]. Отличительной особенностью этого алгоритма является то, что каждое состояние в пространстве поиска задано парой <*cfg*, *interval*>, где *cfg* – конфигурация, описывающая положение агента, т.е. вершину графа, в которой находится агент, а *interval* – безопасный интервал, т.е. промежуток времени, в течение которого агент может находиться в этой вершине не создавая конфликтов. Таким образом одной вершине графа соответствует множество различных состояний, которые могут отличаться безопасными интервалами.

Недостатком алгоритма CCBS можно назвать его относительно низкую вычислительную эффективность. Этот недостаток напрямую связан с его свойством – оптимальностью. Достигается она в том числе за счет того, что на каждом шаге алгоритма из дерева альтернативных решений извлекается решение минимальной стоимости, т.е., по сути, верхний уровень алгоритма работает по принципу алгоритма Дейкстры. Фактически, для того чтобы гарантировать, что найденное решение, не содержащее конфликтов, является оптимальным, необходимо рассмотреть все альтернативные решения меньшей стоимости.

Одним из возможных путей решения этой проблемы является отказ от поиска оптимальных путей и поиск ограниченно-субоптимальных решений вместо них. Для этого необходимо изменить принцип выбора альтернативных решений, извлекаемых из дерева на каждой итерации работы алгоритма. Для этого предлагается использовать подход, используемый в таких алгоритмах как A^*_ε [20] и EES [21]. Ключевой особенностью этих алгоритмов является то, что они используют вторичную эвристику, влияющую на порядок раскрытия вершин в процессе работы алгоритма. Применяя подход этих алгоритмов к алгоритму CCBS, можно получить модификацию, обладающую повышенной вычислительной эффективностью, которая отыскивает ограниченно-субоптимальные решения.

CCBS+FOCAL. В отличие от классических алгоритмов эвристического поиска, таких как A*, в которых используются два списка – OPEN и CLOSED, в алгоритме A^*_ε используется третий список – FOCAL. Он содержит в себе все вершины, f-значение которых не превышает минимальное f-значение из списка OPEN более чем в $(1+\varepsilon)$ раз. Таким образом множество вершин, находящихся в FOCAL, является подмножеством всех вершин, находящихся в OPEN. На каждом шаге алгоритм раскрывает не ту вершину, которая обладает минимальным f-значением в списке OPEN, а ту, что находится в начале списка FOCAL. При этом сам список FOCAL может быть отсортирован по абсолютно любому критерию. Гарантия того, что итоговое найденное решение не превысит стоимость оптимального решения более чем в $(1+\varepsilon)$ раз достигается за счет того, что в список FOCAL попадают только те вершины, которые удовлетворяют ограничению субоптимальности.

Применительно к алгоритмам, использующим подход конфликтно-ориентированного поиска, которые оперируют деревом альтернативных решений на верхнем уровне алгоритма, список FOCAL может использоваться для выбора текущего альтернативного решения на каждом шаге алгоритма. Наиболее подходящим критерием, когда решения сортируются в списке FOCAL, представляется количество конфликтов, которые присутствуют в том или ином альтернативном решении. Предпочитая рассматривать те решения, которые содержат наименьшее число конфликтов, алгоритм может найти решение, не содержащее конфликтов, за меньшее число итераций. Подобный подход был ранее применен в алгоритме ECBS [13].

CCBS+EES. В работе [14] был предложен альтернативный вариант субоптимальной версии алгоритма CBS. В ней вместо подхода алгоритма A^*_ε используется принцип метода EES (англ. Explicit Estimation Search) [21]. Помимо списка FOCAL, аналогичного тому, что используется в алгоритме A^*_ε, EES оперирует еще одним списком – CLEANUP. Он, по сути, является регулярным списком OPEN, в котором все вершины отсортированы в порядке возрастания f- значения. Список OPEN в свою очередь отсортирован по $\hat{f}$- значению, посчитанному с использованием произвольной эвристической функции $\hat{h}$. Список FOCAL содержит подмножество вершин из списка OPEN, удовлетворяющие ограничению $\hat{f}(n) \le (1+\varepsilon)\hat{f}(best_{\hat{f}})$, где $\hat{f}(best_{\hat{f}})$ – $\hat{f}$-значение вершины, обладаю-

щей минимальным $\hat{f}$- значением в списке OPEN. В свою очередь вершины в списке FOCAL отсортированы по собственному произвольному критерию d.

На каждом шаге алгоритма выбирается вершина из начала одного из этих списков. Принцип выбора заключается в следующем:

1) Берется первая вершина из списка FOCAL, если она удовлетворяет ограничению на коэффициент субоптимальности: $if\ f(best_d) \leq (1+\varepsilon) \cdot f(best_f)$.

2) Если условие п.1 не выполняется, то берется первая вершина из списка OPEN, если она удовлетворяет ограничению на коэффициент субоптимальности: $if\ f\left(best_{\hat{f}}\right) \leq (1+\varepsilon) \cdot f(best_f)$.

3) Если п.1 и п.2 не удовлетворяют условиям, то берется первая вершина из списка CLEANUP, отсортированного по f- значениям.

Для вычисления значений функции $\hat{h}$ в оригинальном алгоритме EES используется онлайн-обучение, которое основано на расчете ошибок оценки стоимости и расстояния в процессе поиска решения. Пусть заданы две эвристические функции: h – допустимая, которая оценивает стоимость пути от произвольной вершины до целевой; d – функция, оценивающая расстояние до целевой вершины. Используя их, можно рассчитать ошибку стоимости или расстояния за одну итерацию работы алгоритма:

$$\epsilon_d(n) = d(bc(n)) - (d(n) - 1),$$

$$\epsilon_h(n) = h(bc(n)) - (h(n) - c(n, bc(n)))$$

где $bc(n)$ – лучший потомок n, т.е. тот, который имеет наименьшее $\hat{f}$-значение, а $c(n, bc(n))$ – стоимость перехода от n к $bc(n)$.

Значения ошибок вычисляются на каждой итерации работы алгоритма и на их основе рассчитываются средние значения ошибок $\overline{\epsilon_d}$ и $\overline{\epsilon_h}$, которые используются для расчета функции $\hat{h}$ [24]:

$$\hat{h}(n) = h(n) + \frac{d(n)}{1 - \overline{\epsilon_d}(n)} \cdot \overline{\epsilon_h}(n).$$

Для использования алгоритма EES на верхнем уровне алгоритма CCBS возможно применение формулы, предложенной в работе [17]. В алгоритмах CBS и CCBS на верхнем уровне не используется какая-либо эвристическая функция. Поэтому f-значениями вершин являются стоимости альтернативных решений, а для расчета ошибок используются следующие формулы:

$$\epsilon_h(n) = cost(bc(n)) - cost(n),$$
$$\epsilon_d(n) = h_c(bc(n)) - (h_c(n) - 1),$$

где $cost(n)$ – стоимость альтернативного решения, содержащегося в вершине n, а $h_c(n)$ – количество конфликтов соответствующего решения.

В свою очередь $\hat{h}$ рассчитывается как:

$$\hat{h}(n) = \frac{h_c(n)}{1 - \overline{\epsilon_d}(n)} \cdot \overline{\epsilon_h}(n),$$

т.е. имеет линейную зависимость от числа конфликтов, содержащихся в альтернативном решении вершины n.

Таким образом, в алгоритме CCBS+EES вершины в списке FOCAL отсортированы по возрастанию значения h_c, т.е. по количеству конфликтов, в списке OPEN вершины отсортированы по возрастанию значения $\hat{f}(n) = cost(n) + \hat{h}(n)$, а список CLEANUP – стоимости решения $cost(n)$.

3. Псевдокод алгоритма CCBS и предложенных модификаций

На Рис. 3 представлен псевдокод основного цикла работы алгоритма CCBS. В строках 1-6 происходит инициализация. Сперва осуществляется планирование индивидуальных траекторий агентов (строки 1-3), затем инициализируются ограничения начального решения (строка 4), после чего создается дерево альтернативных решений CT_Tree (строка 5), в которое добавляется начальное решение (строка 6). В основном цикле (строки 7-15) на каждой итерации из дерева решений выбирается лучший кандидат *N*. Если содержащиеся в нем пути не имеют конфликтов, то искомое решение найдено (строки 9-10). В противном случае происходит поиск и выбор конфликта (строка 11). После чего создаются два новых альтернативных решения N_i и N_j, в которых были изменены траектории агентов с индексами i и j, участвующие в рассматриваемом конфликте (строки 12-13). В завершении итерации новые созданные решения добавляются в дерево решений CT_Tree (строка 15).

С точки зрения основного цикла работы алгоритма, предложенные модификации почти не отличаются от оригинального алгоритма CCBS.

Algorithm 1: CCBS Main Loop

```
   input: G = (V, E), Agents, w
1  foreach i from 1 to len(Agents) do
2      π_i ← findPath(G, Agents(i))
3      N.paths(i) ← π_i
4  N.constrains ← ∅
5  create CT_Tree(w)
6  CT_Tree.addNode(N)
7  while CT_Tree.OPEN ≠ ∅ do
8      N ← CT_Tree.getBestNode()
9      if N.paths have no conflicts then
10         return N.paths
11     ⟨(a_i, t_i), (a_j, t_j)⟩ ← FindConflict(N.paths)
12     N_i ← findNewSolution(Agents(i), N, ⟨(a_i, t_i), (a_j, t_j)⟩)
13     N_j ← findNewSolution(Agents(j), N, ⟨(a_i, t_i), (a_j, t_j)⟩)
14     CT_Tree.updateErrors(N_i, N_j) //EES
15     CT_Tree.addNode(N_i); CT_Tree.addNode(N_j)
```

Рис.3. Основной цикл работы алгоритма CCBS

Исключением является строка 14, отвечающая за обновление значений $\overline{\epsilon_d}$ и $\overline{\epsilon_h}$, используемых в алгоритме CCBS+EES. Все основные изменения заключены в процедурах getBestNode и addNode. Процедура addNode различается тем, в какие списки попадает альтернативное решение, так как у субоптимальных модификаций помимо списка OPEN есть дополнительные списки – FOCAL у CCBS+FOCAL или FOCAL и CLEANUP у CCBS+EES. На Рис. 4 представлен псевдокод процедуры getBestNode в зависимости от модификации. Предполагается, что списки OPEN, FOCAL и CLEANUP являются отсортированными по соответствующему критерию.

В случае оригинального алгоритма CCBS процедура выбора лучшего решения тривиальна – из списка OPEN выбирается вершина минимальной стоимости.

В случае CCBS+FOCAL выбирается решение, обладающее минимальным числом конфликтов, так как именно по этому критерию отсортирован список FOCAL. При этом нет необходимости проверять ограничение субоптимальности, поскольку в списке FOCAL содержатся только те решения, которые ему удовлетворяют. Стоит учесть ситуацию, когда после удаления выбранного решения меняется значение минимальной стоимости в списке OPEN. Она возникает в случае, когда текущее выбранное решение находилось в начале не только списка FOCAL, но и OPEN. В этом случае необходимо обновить список FOCAL, так как возможно, что в него можно добавить большее число решений, которые теперь удовлетворяют ограничению на субоптимальность (строки 3-4 Алгоритма 3).

Algorithm 2: CCBS getBestNode

```
1  bestN ← OPEN.front()
2  delete bestN from OPEN
3  return bestN
```

Algorithm 3: CCBS+FOCAL getBestNode

```
1  bestN ← FOCAL.front()
2  delete bestN from FOCAL and OPEN
3  if minimal cost in OPEN has changed then
4      update FOCAL
5  return bestN
```

Algorithm 4: CCBS+EES getBestNode

```
1  bestCost ← CLEANUP.front().cost
2  if FOCAL.front().cost ≤ bestCost * w then
3      bestN ← FOCAL.front()
4  else if OPEN.front().cost ≤ bestCost * w then
5      bestN ← OPEN.front()
6  else
7      bestN ← CLEANUP.front()
8  delete bestN from CLEANUP and OPEN
9  if bestN ∈ FOCAL then
10     delete bestN from FOCAL
11 if minimal f̂ in OPEN has changed then
12     update FOCAL
13 return bestN
```

Рис.4. Принцип работы процедуры getBestNode в зависимости от модификации алгоритма CCBS

В свою очередь алгоритм CCBS+EES выбирает решение из начала списка FOCAL, если оно удовлетворяет ограничению на субоптимальность (строки 2-3 Алгоритма 4). В противном случае выбирается решение из начала списка OPEN, если оно удовлетворяет ограничению на субоптимальность (строки 4-5 Алгоритма 4). В крайнем случае берется решение из начала списка CLEANUP, которое гарантированно удовлетворяет ограничению на субоптимальность, т.к. этот список отсортирован именно по критерию стоимости (строки 6-7 Алгоритма 4). Затем решение удаляется из соответствующих списков (строки 8-9 Алгоритма 4) и в случае изменения минимального $\hat{f}$-значения происходит обновление списка FOCAL, так как теперь в него могут быть добавлены дополнительные решения (строки 11-12 Алгоритма 4).

4. Теоретические свойства

Предложенные модификации алгоритма CCBS, а именно CCBS+FOCAL и CCBS+EES, обладают свойствами ограниченной субоптимальности, т.е. гарантируют, что отыскиваемые решения имеют стоимость, не превышающую стоимость оптимального решения более чем в w раз. Доказательства этого свойства основано на следующем утверждении.

Утверждение 1. Значение минимальной стоимости решения в списке OPEN монотонно не убывает.

Доказательство. Предположим, что Утверждение 1 является ложным. Следовательно, существует итерация алгоритма i, на которой стоимость минимального решения в списке OPEN была $cost_{min_i}$, а на итерации $i+1$ – $cost_{min_{i+1}}$, при этом $cost_{min_{i+1}} < cost_{min_i}$. Стоимость любого решения, содержащегося в дереве решений на начало итерации i, не может быть меньше $cost_{min_i}$. Так как на каждой итерации алгоритма из списка OPEN извлекается одно решение и добавляется не более двух новых, получается, что стоимостью $cost_{min_{i+1}}$ обладает, по крайней мере, одно из новых решений, добавленных в список OPEN на итерации i. Таким образом, получается, что, по крайней мере, один из потомков, сгенерированных на итерации i, обладает стоимостью меньшей, чем его родитель. Решение потомка отличается от решения родителя только траекторией того агента, на которого было наложено дополнительное ограничение. При этом траектория, спланированная с дополнительным ограничением, может иметь стоимость меньшую, чем спланированная без него только в том случае, если алгоритм планирования нижнего уровня не обладает свойством оптимальности. Что вызывает противоречие, так как алгоритм SIPP, используемый в алгоритме CCBS и предложенных модификациях, этим свойством обладает. В результате возникшего противоречия Утверждение 1 является истинным.

Утверждение 2. Алгоритм CCBS+FOCAL является ограниченно-субоптимальным, т.е. гарантирует, что стоимость найденного им решения не превышает стоимость оптимального решения более чем в w раз, где w – коэффициент субоптимальности.

Доказательство. Доказательство этого утверждения основано на том, что, во-первых, алгоритм CCBS является оптимальным, т.е. минимальная стоимость решения, содержащегося в списке OPEN, не превышает стоимость оптимального решения. Во-вторых, в список FOCAL попадают только те альтернативные решения, стоимость которых удовлетворяет ограничению на субоптимальность. Проверка на ограничение осуществляется в момент, когда решение попадает в список FOCAL. При этом на любой последующей итерации все решения, содержащиеся в списке FOCAL, продолжают удовлетворять ограничению на субоптимальность. Доказательство этого утверждения не требуется, так как оно является прямым следствием Утверждения 1. Таким образом, какое бы решение не было извлечено из списка FOCAL на любой итерации алгоритма, оно удовлетворяет ограничению на субоптимальность. Следовательно, итоговое решение, извлеченное на последней итерации работы алгоритма, является ограниченно-субоптимальным.

Утверждение 3. Алгоритм CCBS+EES является ограниченно-субоптимальным, т.е. гарантирует, что стоимость найденного им решения не превышает стоимость оптимального решения более чем в w раз, где w – коэффициент субоптимальности.

Доказательство. Как и в случае с алгоритмом CCBS+FOCAL, доказательство этого утверждения основано в первую очередь на том, что алгоритм CCBS является оптимальным,

т.е. значение минимальной стоимости решения, содержащегося в списках OPEN и CLEANUP, не превышает стоимость оптимального решения. В отличие от алгоритма CCBS+FOCAL, в алгоритме CCBS+EES списки OPEN и FOCAL могут содержать решения, которые не удовлетворяют ограничению на субоптимальность. Однако проверка на ограничение осуществляется на каждой итерации работы алгоритма непосредственно в момент выбора текущего лучшего решения. Если решение не удовлетворяет ограничению, то оно не извлекается из списка. В крайнем случае, если оба лучших решения в списках OPEN и FOCAL не удовлетворяют ограничению на субоптимальность, то в качестве лучшего выбирается то из списка CLEANUP, которое отсортировано по стоимости, т.е. фактически выбирается решение минимальной стоимости. Таким образом, на любой итерации алгоритма в качестве текущего лучшего решения может быть выбрано только то, которое удовлетворяет ограничению на субоптимальность. Следовательно, итоговое решение, извлеченное на последней итерации работы алгоритма, является ограниченно-субоптимальным.

5. Экспериментальные исследования

Для проведения модельных экспериментальных исследований были написаны программные реализации обеих модификаций алгоритма CCBS – CCBS+FOCAL и CCBS+EES. Экспериментальные исследования производились на картах и заданиях, взятых из открытой коллекции MovingAI [22]. Всего было использовано 4 карты (Рис. 5): empty-16-16 (16x16 вершин), room-64-64-8 (64x64 вершины), warehouse-10-20-10-2-2 (172x84 вершины), den520d (256x257 вершин).

Для каждой из карт даны 25 сценариев, содержащих стартовые и целевые положения для сотен агентов. Решить их полностью за ограниченное время не представляется возможным, поэтому тестирование каждого сценария проводилось по схеме постепенного увеличения числа агентов. Изначально берутся лишь первые 2 агента. Если алгоритм смог решить это задание, то добавляется следующий агент из сценария, и процесс повторяется. Таким образом количество агентов в задании постепенно увеличивается, а вместе с их количеством повышается и сложность задания. Процесс тестирования одного сценария продолжается до тех пор, пока алгоритм может решить задание в установленное время.

Во время проведения экспериментальных исследований агентам были разрешены перемещения, как в ортогонально- так и диагонально-смежные вершины. Все агенты считались гетерогенными, движущимися с постоянной скоростью и имеющими радиус $r = \sqrt{2}/4$.

Результаты предварительного тестирования показали, что для получения более наглядных результатов необходимо использовать различные значения коэффициента субоптимальности в зависимости от размера карты. Это связано с тем, что на картах большого размера, таких как *den520d* и *warehouse*, агенты преодолевают большие расстояния и итоговая стоимость решения, не содержащая конфликтов, в процентном соотношении мало отличается от начального решения, полученного путем независимого планирования траектории каждого агента. В связи с этим, для карт *den520d* и *warehouse* были протестированы следующие значения w: 1.001, 1.003, 1.005, 1.01 и 1.03, а для карт *16x16_empty* и *rooms* – 1.01, 1.03, 1.05, 1.1, 1.25.

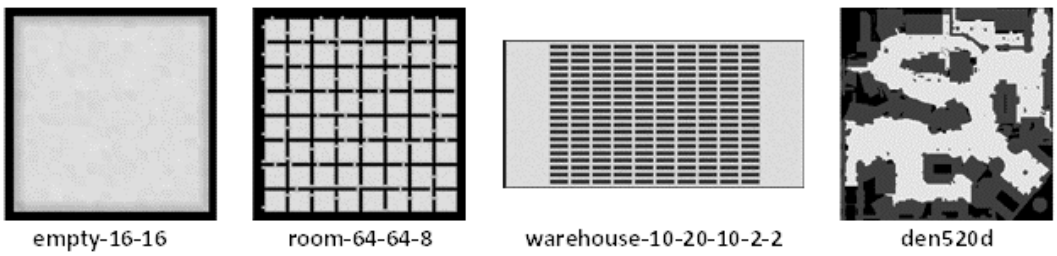

Рис.5. Графическое представление карт empty-16-16, room-64-64-8, warehouse-10-20-10-2-2 и den520d, использовавшиеся для проведения первой серии экспериментов

Для оценки вычислительной эффективности работы алгоритмов были проанализированы такие показатели как общее число решенных заданий и среднее число итераций работы алгоритма. Было также проведено сравнение качества найденных решений и оценка их фактической субоптимальности.

На Рис. 6 приведены показатели общего числа решенных заданий каждым из алгоритмов. Числа над столбцами показывают абсолютные значения, а шкала – процентное соотношение относительно числа заданий, решенных оптимальной версией алгоритма CCBS. Как показывают результаты экспериментов, обе субоптимальные версии алгоритма CCBS способны решать значительно более сложные задания, содержащие до двух и более раз большее число агентов. При этом нельзя однозначно сказать какая из версий работает лучше, так как на разных картах результаты демонстрируют разные тенденции.

Стоит отметить, что дальнейшее увеличение значения коэффициента *w* не провидит к существенному увеличению числа решаемых заданий, т.е. полученные значения числа решаемых заданий близки к максимальным, что можно получить на данных картах, сценариях и соответствующих параметрах. При этом не во всех случаях увеличение значения коэффициента субоптимальности приводит к увеличению числа заданий, решаемых алгоритмом. В отдельных сценариях наблюдались случаи, когда алгоритм, использующий более высокое значение *w*, не может найти решение в установленный лимит времени, в то время как версия алгоритма с меньшим значением находит его за доли секунды. Такое поведение объясняется тем, что в зависимости от значения коэффициента субоптимальности алгоритм может выбирать для раскрытия разные ветви дерева решений и часть из них либо требуют наложения слишком большого числа различных ограничений, что приводит к необходимости выполнения большого числа итераций, либо в принципе не способны привести алгоритм к решению, не содержащему конфликтов.

В Табл. 1 представлено среднее число итераций работы алгоритма CCBS. Для усреднения были взяты те задания, которые были успешно решены всеми версиями алгоритма. При этом из общей базы были исключены слишком простые задания, т.е. те, которые не содержали конфликтов в начальном решении, так как поиск их решения невозможно ускорить. Из усреднения были также исключены задания, в которых оптимальный алгоритм CCBS совершал более 10000 итераций. Таких заданий на каждой карте было порядка 2-5% от общего числа решенных заданий и их можно считать статистическими выбросами. Данные в таблице показывают, что на поиск решений тех заданий, которые способен решить оптимальный

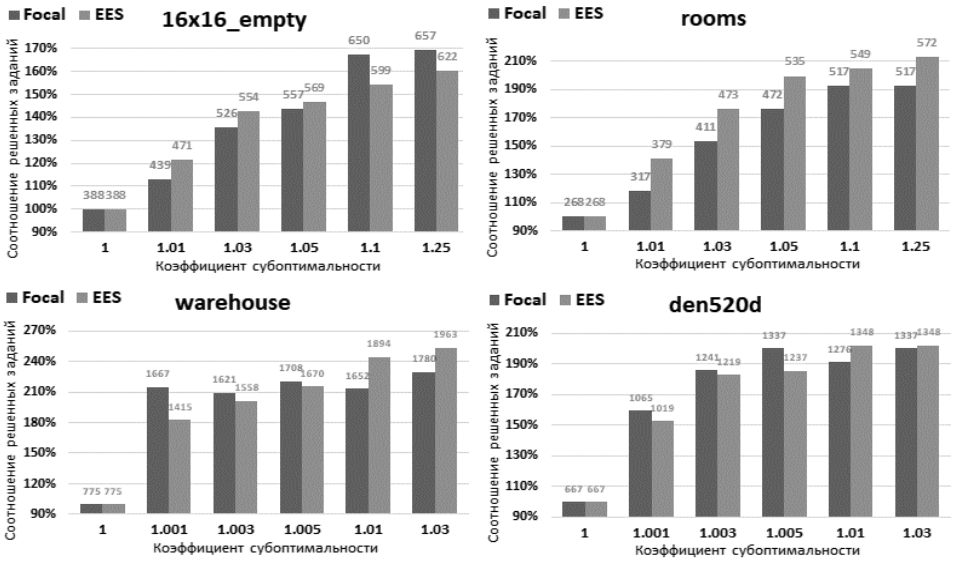

Рис. 6. Общее число и соотношение решенных заданий на картах 16x16_empty, rooms, warehouse и den520d

алгоритм CCBS в течение установленного лимита времени, субоптимальным версиям алгоритма требуется на порядок меньшее число итераций. При этом разница между алгоритмами CCBS+FOCAL и CCBS+EES в большинстве случаев незначительная.

Далее был проанализирован показатель фактического превышения стоимости решений, найденных субоптимальными версиями алгоритма, по сравнению с оптимальными решениями, в зависимости от используемого коэффициента (Рис. 7). Как и в случае с анализом числа раскрытых вершин, для подсчета использовалась общая база заданий, решенных всеми версиями алгоритма. При этом из усреднения были исключены задания, которые не содержали конфликтов в начальном решении. Для наглядной демонстрации полученного распределения была использована «ящичковая диаграмма» или «ящик с усами». Границами ящика являются первый и третий квартили линия в середине ящика – медиана, концы усов – края статистически значимой выборки. Точки, показанные на графике – выбросы, т.е. конкретные задания, на которых были достигнуты соответствующие превышения стоимости. График показывает, что в подавляющем большинстве случаев фактическое увеличение стоимости найденного решения не превышает 1% вне зависимости от используемого коэффициента. При этом есть ряд заданий, в которых алгоритмы могут существенно превысить это значение и приблизиться к границе установленного порога по субоптимальности.

Результаты проведенных экспериментов показали значительное повышение вычислительной эффективности алгоритма CCBS, при переходе от поиска гарантированно-

Табл. 1. Среднее число итераций работы верхнего уровня алгоритма на картах 16x16_empty, rooms, warehouse и den520d

	CCBS	CCBS+FOCAL					CCBS+EES				
w	1.0	1.01	1.03	1.05	1.1	1.25	1.01	1.03	1.05	1.1	1.25
16x16_empty	134.8	19.2	12.7	12.7	12.2	12.2	15.4	12.9	12.7	12.7	12.7
rooms	157.3	31.0	13.8	13.1	13.3	13.3	21.1	18.5	19.3	14.3	13.1
	CCBS	CCBS+FOCAL					CCBS+EES				
w	1.0	1.001	1.003	1.005	1.01	1.03	1.001	1.003	1.005	1.01	1.03
warehouse	122.7	12.5	24.3	32.8	16.8	50.6	15.3	41.9	32.8	55.6	55.6
den520d	168.3	9.8	9.8	9.8	13.0	10.1	10.5	9.8	9.8	10.0	10.0

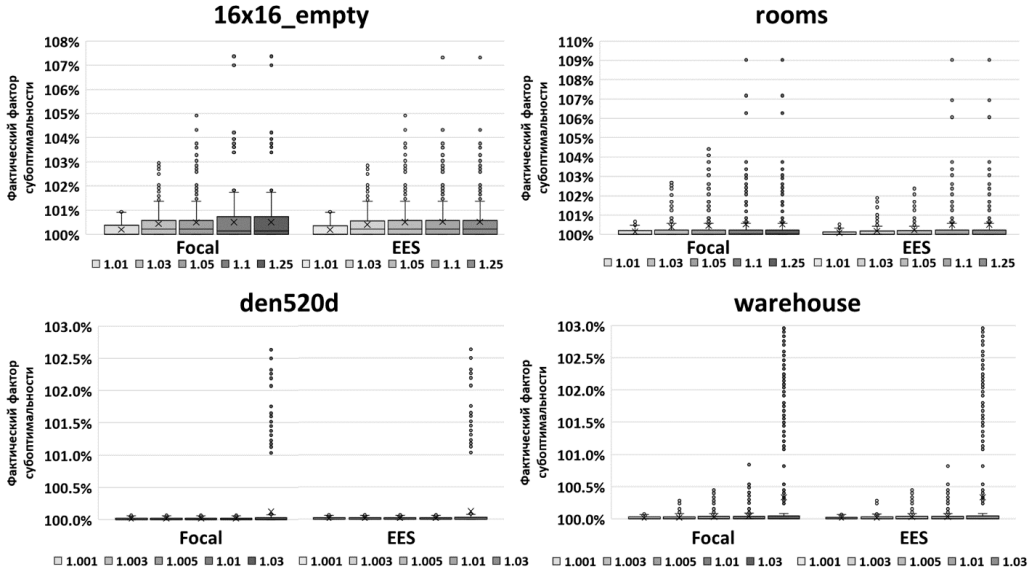

Рис. 7. Фактическое превышение стоимости решений на картах 16x16_empty, rooms, warehouse и

оптимальных решений к поиску ограниченно-субоптимальных. В большинстве случаев, даже используя коэффициент 1.01, что эквивалентно максимально возможному превышению стоимости решения в 1%, алгоритмы CCBS+FOCAL и CCBS+EES способны решать задания, содержащие до двух раз большее число агентов, что говорит о существенно возросшей вычислительной эффективности. С точки зрения сравнения двух различных субоптимальных версий алгоритма между собой, то проведенные экспериментальные исследования показали, что эффективность их работы и качество отыскиваемых решений в большинстве случаев находятся на одном уровне. При этом наблюдается тенденция, что при малых значениях коэффициента субоптимальности, предпочтительнее использовать CCBS+EES.

Заключение

В статье рассматривалась задача многоагентного планирования с учетом возможности совершения действий произвольной продолжительности. Для решения этой задачи рассматривался алгоритм Continuous-time Conflict Based Search (CCBS), использующий подход конфликтно-ориентированного поиска. Были предложены две модификации этого алгоритма, позволяющие находить ограниченно-субоптимальные решения – CCBS+FOCAL и CCBS+EES. Результаты проведенных модельных экспериментальных исследований продемонстрировали существенное увеличение вычислительной эффективности работы алгоритма. В сравнении с оригинальным алгоритмом CCBS, предложенные модификации способны решать задания, содержащие в 2-3 раза большее число агентов. При этом на поиск решений тех заданий, которые успешно решаются алгоритмом CCBS в течение установленного лимита времени, CCBS+FOCAL и CCBS+EES затрачивают на порядок меньшее число итераций, что соответствующим образом сокращает время их работы.

Одним из дальнейших направлений работ может стать разработка anytime-версии алгоритма CCBS, т.е. такой версии, которая за ограниченное время будет находить решение, а затем, с увеличением доступного лимита времени, пытаться улучшить качество этого решения, при этом повторно используя результаты предыдущих вычислений.

Литература

1. Бухвалов О.Л., Городецкий В.И., Карасев О.В. и др. Производственная логистика: Стратегическое планирование, прогнозирование и управление конфликтами // Известия ЮФУ. 2012. №3. С. 209–218.
2. Иванов А.М. Разработка системы межобъектного взаимодействия интеллектуальных транспортных средств // Известия ВолгГТУ. Серия «Наземные транспортные системы». Вып. 7: межвуз. сб. науч. ст./ВолгГТУ. Волгоград. 2013. № 21 (124) . С. 74-77.
3. Ронжин А.Л., Ву Д.К., Нгуен В.В., Соленая О.Я. Концептуальная и алгоритмические модели совместного функционирования роботизированной платформы и набора БЛА при выполнении аграрных операций // IV всероссийский научно-практический семинар «Беспилотные транспортные средства с элементами искусственного интеллекта». Труды семинара. Казань. 2017. С. 183-192.
4. Мотиенко А.И. Планирование тактической траектории движения автоматизированных робототехнических средств при ликвидации последствий чрезвычайных ситуаций// Научные ведомости Белгородского государственного университета. Серия: Экономика. Информатика. 2016. Т. 37. № 2 (223). С. 139-143.
5. Erdmann M., Lozano-Pérez T. On multiple moving objects // Algorithmica. T.2. 1987. P. 1419–1424.
6. Standley T. Finding optimal solutions to cooperative pathfinding problems //Proceedings of the AAAI Conference on Artificial Intelligence. 2010. T. 24. №. 1. P.173-178.
7. Hart P.E., Nilsson N.J., Raphael B. A formal basis for the heuristic determination of minimum cost paths. // IEEE transactions on Systems Science and Cybernetics 4(2). 1968. P. 100–107.
8. Sharon G., Stern R., Felner A., Sturtevant N.R. Conflict-based search for optimal multi-agent pathfinding // Artificial Intelligence. 2015. №. 219. P. 40–66.
9. Wagner G., Choset H. M*: A complete multirobot path planning algorithm with performance bounds //2011 IEEE/RSJ international conference on intelligent robots and systems. IEEE. 2011. P. 3260-3267.
10. Sharon G., Stern R., Goldenberg M., Felner A. The increasing cost tree search for optimal multi-agent pathfinding. // Artificial Intelligence. 2013. T. 195. P. 470–495.
11. Boyarski E., Felner A., Stern R., Sharon G., Betzalel O., Tolpin D., Shimony E. ICBS: The improved conflict-based search algorithm for multi-agent pathfinding //Proceedings of the Eighth International Symposium on Combinatorial Search (SoCS-2015). 2015. P. 223-225.
12. Felner A., Li J., Boyarski E., Ma H., Cohen L., Kumar T.S., Koenig S. Adding heuristics to conflict-based search for multi-agent path finding //Proceedings of the 28th International Conference on Automated Planning and Scheduling. 2018. P.83–87.
13. Barer M., Sharon G., Stern R., Felner A. Suboptimal variants of the conflict-based search algorithm for the multi-

agent pathfinding problem //Seventh Annual Symposium on Combinatorial Search. 2014.
14. Li J., Ruml W., Koenig S. EECBS: Bounded-Suboptimal Search for Multi-Agent Path Finding // Proceedings of the 35th AAAI Conference on Artificial Intelligence. 2021. P. 12353–12362.
15. Yakovlev K., Andreychuk A. Any-Angle Pathfinding for Multiple Agents Based on SIPP Algorithm //Proceedings International Conference on Automated Planning and Scheduling, ICAPS. 2017. P. 586-593.
16. Walker T.T., Sturtevant N.R., Felner, A. Extended increasing cost tree search for non-unit cost domains // Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI-2018). 2018. P. 534–540.
17. Cohen L., Uras T., Kumar T. S., Koenig S. Optimal and bounded-suboptimal multi-agent motion planning. In Proceedings of the 12th Symposium on Combinatorial Search (SoCS 2019). 2019. P. 44-51.
18. Guy S. J., Karamouzas I. Guide to anticipatory collision avoidance //Game AI Pro 2: Collected Wisdom of Game AI Professional – AK Peters/CRC Press. 2015. T. 2. P. 195-207.
19. Phillips M., Likhachev M. SIPP: Safe interval path planning for dynamic environments // 2011 IEEE International Conference on Robotics and Automation. IEEE.2011. P. 5628–5635.
20. Pearl J., Kim, J. H. Studies in Semi-Admissible Heuristics. //IEEE Transaction on Pattern Analysis and Machine Intelligence. 1982. T.4. P.392–399.
21. Thayer J. T., Ruml W. Bounded Suboptimal Search: A Direct Approach Using Inadmissible Estimates // Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-2011). 2011. P.674–679.
22. Sturtevant N.R. Benchmarks for grid-based pathfinding // IEEE Transactions on Computational Intelligence and AI in Games 4(2). 2012. P. 144–148.

Андрейчук Антон Андреевич. Ведущий исследователь. Автономная некоммерческая организация «Институт искусственного интеллекта». Ассистент. Российский университет дружбы народов. Области исследований: планирование траектории, эвристический поиск, многоагентные системы. E-mail: andreychuk@mail.com

Effective Bounded-Suboptimal Algorithm of Solving Multi-Agent Pathfinding Problem

A. A. Andreychuk

Peoples Friendship University of Russia (RUDN University), Moscow, Russia

Abstract: The paper considers the problem of finding a combination of collision-free trajectories for a set of agents capable of performing actions of arbitrary duration. To solve this problem, two bounded-suboptimal modifications of the continuous-time conflict-based search algorithm are proposed. The results of the carried out model experimental studies have demonstrated the high computational efficiency of the proposed modifications.

Keywords: path planning, pathfinding, grid, graph, multi-agent systems, MAPF.

DOI 10.14357/20718594220106

References

1. Bukhvalov, O.L., Gorodetskij, V.I., Karasev, O.V. et al. Proizvodstvennaya logistika: Strategicheskoe planirovanie, prognozirovanie i upravlenie konfliktami [Industrial Logistics: Strategic Planning, Forecasting and Conflict Management] // Izvestiya YUFU [SFU Tidings]. 2012. V. 3. P. 209–218.
2. Ivanov, A.M. Razrabotka sistemy mezhob"ektnogo vzaimodejstviya intellektual'nykh transportnykh sredstv [Development of a system of interobject interaction of intelligent vehicles] // Izvestiya VolgGTU. Seriya «Nazemnye transportnye sistemy» [VolgSTU Tidings. Series "Land transport systems."]. 2013. V. 7. №. 21(124). P. 74–77.
3. Ronzhin, A.L., Vu, D.K., Nguen, V.V., Solenaya, O.Ya. Kontseptual'naya i algoritmicheskie modeli sovmestnogo funktsionirovaniya robotizirovannoj platformy i nabora BLA pri vypolnenii agrarnykh operatsij [Conceptual and algorithmic models of the joint operation of a robotic platform and a set of UAVs in the performance of agrarian operations] // IV vserossijskij nauchno-prakticheskij seminar "Bespilotnye transportnye sredstva s elementami iskusstvennogo intellekta". Trudy seminara. Kazan'. [IV All-Russian Scientific and Practical Seminar "Unmanned Vehicles with Artificial Intelligence Elements". Proceedings of the seminar. Kazan]. 2017. P. 183 -192.
4. Motienko, A.I. Planirovanie takticheskoj traektorii dvizheniya avtomatizirovannykh robototekhnicheskikh sredstv prilikvidatsii posledstvij chrezvychajnykh situatsij [Planning of the tactical trajectory of the movement of automated robotic means during the liquidation of the consequences of emergency situations] // Nauchnye vedomosti Belgorodskogo gosudarstvennogo universiteta. Seriya: EHkonomika. Informatika. [Scientific bulletins of the Belgorod State University. Series: The Economy. Computer science.]. 2016. V. 37. №. 2(223). P.139–143.
5. Erdmann M., Lozano-Pérez T. On multiple moving objects // Algorithmica. 1987. V. 2. P. 1419–1424.

6. Standley T. Finding optimal solutions to cooperative pathfinding problems //Proceedings of the AAAI Conference on Artificial Intelligence. 2010. V. 24. №. 1. P.173-178.
7. Hart P.E., Nilsson N.J., Raphael B. A formal basis for the heuristic determination of minimum cost paths. // IEEE transactions on Systems Science and Cybernetics 4(2), – 1968. – P. 100–107.
8. Sharon G., Stern R., Felner A., Sturtevant N.R. Conflict-based search for optimal multi-agent pathfinding // Artificial Intelligence. 2015. №. 219. P. 40–66.
9. Wagner G., Choset H. M*: A complete multirobot path planning algorithm with performance bounds //2011 IEEE/RSJ international conference on intelligent robots and systems. IEEE. 2011. P. 3260-3267.
10. Sharon G., Stern R., Goldenberg M., Felner A. The increasing cost tree search for optimal multi-agent pathfinding. // Artificial Intelligence. V. 195. 2013. P. 470–495.
11. Boyarski E., Felner A., Stern R., Sharon G., Betzalel O., Tolpin D., Shimony E. ICBS: The improved conflict-based search algorithm for multi-agent pathfinding //Proceedings of the Eighth International Symposium on Combinatorial Search (SoCS-2015). 2015. P.223-225.
12. Felner A., Li J., Boyarski E., Ma H., Cohen L., Kumar T. S., Koenig S. Adding heuristics to conflict-based search for multi-agent path finding //Proceedings of the 28th International Conference on Automated Planning and Scheduling. 2018. P.83–87.
13. Barer M., Sharon G., Stern R., Felner A. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem //Seventh Annual Symposium on Combinatorial Search. 2014.
14. Li J., Ruml W., Koenig S. EECBS: Bounded-Suboptimal Search for Multi-Agent Path Finding // Proceedings of the 35th AAAI Conference on Artificial Intelligence. 2021. P. 12353–12362.
15. Yakovlev K., Andreychuk A. Any-Angle Pathfinding for Multiple Agents Based on SIPP Algorithm //Proceedings International Conference on Automated Planning and Scheduling, ICAPS. 2017. P. 586-593.
16. Walker T.T., Sturtevant N.R., Felner, A. Extended increasing cost tree search for non-unit cost domains // Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI-2018). 2018. P. 534–540.
17. Cohen L., Uras T., Kumar T. S., Koenig S. Optimal and bounded-suboptimal multi-agent motion planning // Proceedings of the 12th Symposium on Combinatorial Search (SoCS 2019). 2019. P. 44-51.
18. Guy S. J., Karamouzas I. Guide to anticipatory collision avoidance //Game AI Pro 2: Collected Wisdom of Game AI Professional – AK Peters/CRC Press. 2015. V. 2. P. 195-207.
19. Phillips M., Likhachev M. SIPP: Safe interval path planning for dynamic environments // 2011 IEEE International Conference on Robotics and Automation. IEEE. 2011. P. 5628–5635.
20. Pearl J., Kim, J. H. Studies in Semi-Admissible Heuristics. //IEEE Transaction on Pattern Analysis and Machine Intelligence. 1982. V. 4. P. 392–399.
21. Thayer J. T., Ruml W. Bounded Suboptimal Search: A Direct Approach Using Inadmissible Estimates // Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-2011). 2011. P.674–679.
22. Sturtevant N.R. Benchmarks for grid-based pathfinding // IEEE Transactions on Computational Intelligence and AI in Games 4(2). 2012. P. 144–148.

Andreychuk Anton A. Leading researcher, Artificial Intelligence Research Institute. Assistant of the department, Peoples' Friendship University of Russia (RUDN University). Research areas: path planning, heuristic search, multi-agent systems. E-mail: andreychuk@mail.com