

Алгоритмы конфликтно-ориентированного поиска для задачи многоагентного планирования (CBS, CBS+PC)

Проблема

Задача нахождения пути для одного агента на некоторой карте - задача нахождения пути между двумя вершинами в графе. Задача многоагентного планирования (Multi-agent pathfinding, MAPF) - обобщение этой задачи для $k > 1$ агентов. В случае такой задачи имеется карта и множество агентов, для каждого из которых определены стартовая и целевая позиции. Проблема заключается в нахождении путей для каждого агента так, чтобы избежать столкновений между ними. Алгоритмы для ее решения можно разделить на оптимальные и субоптимальные. В данной работе рассматривается поиск оптимального решения для задачи многоагентного планирования.

Формальная постановка задачи

В задаче производится работа с ориентированным графом $G(V, E)$ и с k агентами (a_1, \dots, a_k) . У каждого агента a_i имеется стартовая вершина $start_i \in V$ и целевая вершина $goal_i \in V$. Время дискретизируется на точки t_0, t_1 и т.д. В момент времени t_0 агент a_i находится в $start_i$.

За соответствующий промежуток времени агент может совершить переход в какую-либо соседнюю вершину либо оставаться в текущей вершине (совершать действие «ожидание»).

Главное ограничение в MAPF заключается в том, что в каждой вершине в текущий момент времени может находиться не более одного агента. Также можно ввести ограничение на переход агентов по одному и тому же ребру между соответствующими моментами времени (в данной работе это ограничение также учитывается). Конфликт - ситуация, когда ограничение нарушается.

Решением MAPF является множество неконфликтующих путей - по одному для каждого агента - где путь для агента a_i - последовательность действий, приводящая агента a_i из $start_i$ в $goal_i$.

Для решения этой задачи производится минимизация глобальной общей функции стоимости. Эта функция получается суммированием по всем агентам числа шагов, которые требуются, чтобы из стартовой точки прийти в целевую и не покидать ее. MAPF можно разделить на 2 группы: распределенные и централизованные. В распределенной постановке у каждого агента есть своя вычислительная мощность и могут быть использованы разные виды коммуникации (переда сообщений, трансляция и т. д.). В централизованной постановке подразумевается наличие некоторого одного ресурса, который должен найти решение для всех агентов. В этой работе рассматривается централизованный вариант MAPF.

Вход и выход

Вход:

1. Ориентированный граф $G(V, E)$. Вершины графа - возможные положения агентов, ребра - возможные перемещения между местами.
2. k агентов, обозначенные a_1, a_2, \dots, a_k . У каждого агента a_i имеется стартовая вершина $start_i \in V$ и целевая вершина $goal_i \in V$. Время представлено как точки t_0, t_1 и т.д. В момент времени t_0 агент a_i находится в $start_i$.

Выход: множество неконфликтующих путей - по одному для каждого агента.

Принцип работы алгоритма конфликтно-ориентированного поиска (CBS)

Здесь и далее будут использоваться следующие понятия и утверждения:

- Слово «путь» будет использоваться в контексте одного агента.
- Слово «решение» будет обозначать множество k путей для данного множества k агентов.
- Ограничение - набор (a_i, v, t) , где агенту a_i запрещено занимать вершину v в момент времени t .
- Состоятельный путь для агента a_i - путь, который удовлетворяет всем ограничениям для этого агента.
- Состоятельное решение - решение, которое состоит из путей таких, что каждый путь для любого агента a_i состоятелен.
- Конфликт - набор (a_i, a_j, v, t) , где агенты a_i и a_j занимают вершину v в момент времени t .
- Решение допустимо, если все входящие в него пути не содержат конфликтов.
- Состоятельное решение может быть недопустимо, если, несмотря на состоятельность всех входящих в него путей, эти пути все равно содержат конфликты.

Ключевая идея конфликтно-ориентированного поиска (conflict-based search, CBS) - увеличивать множество ограничений и находить пути, которые согласуются с ними. CBS работает на двух уровнях: низком и высоком. На высоком уровне ищутся конфликты, добавляются ограничения. На низком уровне ищутся индивидуальные пути для агентов, которые состоятельны по отношению к новым добавляемым конфликтам.

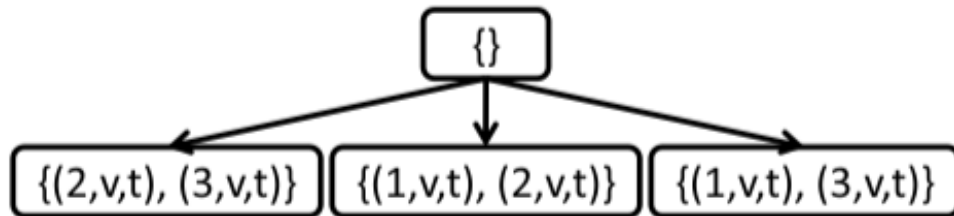
В <http://www.bgu.ac.il/~felner/2015/CBSjur.pdf> приводится доказательство оптимальности алгоритма.

Высокий уровень

На данном уровне производится поиск по дереву, которое называется деревом ограничений. Каждая вершина дерева N включает в себя:

- множество ограничений (корень дерева содержит пустое множество ограничений);
- решение (каждый путь состоятелен, пути ищутся на низком уровне);
- стоимость текущего решения (сумма стоимостей всех индивидуальных путей).

Вершина N целевая, если ее решение допустимо. На высоком уровне выполняется поиск лучшей вершины (вершины отсортированы по стоимостям). В этой работе в случае ничьих выбирается вершина, у которой решение содержит меньше конфликтов; дальнейшие ничьи разрешаются по принципу first in - first out. В соответствии с набором ограничений вершины выполняется поиск на низком уровне; этот поиск возвращает кратчайший путь для каждого агента (с учетом его ограничений). Когда для каждого агента найден состоятельный путь, эти пути валидируются на наличие конфликтов (рассматриваются все местоположения агентов в каждый момент времени). Если конфликтов нет, текущий N - целевой, возвращается текущее решение. Если обнаружен конфликт $C = (a_1 \dots a_k, v, t)$ для k ($k \geq 2$) агентов, валидация останавливается и N объявляется нецелевым. В случае конфликта k агентов ($k \geq 2$) генерируются k потомков N с $k-1$ ограничением (в каждом потомке одному из k агентов позволяется занимать эту вершину в момент t).



Конфликты ребер (ситуации, когда два агента меняются местами, то есть переходят по ребру в противоположных направлениях за один и тот же промежуток времени) обрабатываются аналогично.

Алгоритм 2 представляет собой псевдокод для CBS (MA-CBS не используется).

Algorithm 2: High level of CBS (and MA-CBS).

```

Input: MAPF instance
1   $Root.constraints = \emptyset$ 
2   $Root.solution = \text{find individual paths by the low level}()$ 
3   $Root.cost = SIC(Root.solution)$ 
4  insert  $Root$  to OPEN
5  while OPEN not empty do
6       $P \leftarrow$  best node from OPEN // lowest solution cost
7      Validate the paths in  $P$  until a conflict occurs.
8      if  $P$  has no conflict then
9          return  $P.solution$  //  $P$  is goal
10      $C \leftarrow$  first conflict  $(a_i, a_j, v, t)$  in  $P$ 
11     if  $shouldMerge(a_i, a_j)$  // Optional, MA-CBS only then
12          $a_{i,j} = merge(a_i, a_j, v, t)$ 
13         Update  $P.constraints$ (external constraints).
14         Update  $P.solution$  by invoking low level( $a_{i,j}$ )
15         Update  $P.cost$ 
16         if  $P.cost < \infty$  // A solution was found then
17             Insert  $P$  to OPEN
18         continue // go back to the while statement
19     foreach agent  $a_i$  in  $C$  do
20          $A \leftarrow$  new node
21          $A.constraints \leftarrow P.constraints + (a_i, v, t)$ 
22          $A.solution \leftarrow P.solution$ 
23         Update  $A.solution$  by invoking low level( $a_i$ )
24          $A.cost = SIC(A.solution)$ 
25         if  $A.cost < \infty$  // A solution was found then
26             Insert  $A$  to OPEN

```

Низкий уровень

На низкий уровень подается агент со своими ограничениями, для которого ищется оптимальный путь, который удовлетворял бы всем ограничениям, при этом наличие остальных агентов игнорируется. В данной работе для этого применяется алгоритм A^* .

CBS с приоритизацией конфликтов (CBS+PC)

Проблема

CBS чувствителен к тому, на каком конфликте останавливать валидацию и делать разделение (то есть порождать потомков N). Базовый CBS выбирает первый найденный конфликт, что может приводить к медленной работе (алгоритм «застревает» в области, где $N.cost$ схожи и порождать большое количество лишних вершин дерева конфликтов.

Решение: приоритизация конфликтов (PC)

Для решения проблемы вводится приоритизация конфликтов.

Рассматриваются 3 типа конфликтов для 2 агентов:

- Кардинальный конфликт. Конфликт (a_1, a_2, v, t) называется кардинальным для вершины N дерева ограничений, если добавление любого из двух ограничений, полученных из этого конфликта, к N и выполнению нового поиска на низком уровне для агента a_i приводит к увеличению стоимости его пути по сравнению с исходным значением в N . Эквивалентное определение: конфликт кардинален, если все состоятельные пути для a_1 и a_2 содержат вершину v в момент времени t .
- Полукардинальный конфликт. Полукардинальный конфликт определяется аналогично, но к увеличению стоимости приводит только одно из двух ограничений.
- Некардинальный конфликт. Некардинальный конфликт определяется аналогично, но ни одно из двух ограничений не приводит к увеличению стоимости.

MDD - многозначная диаграмма выбора - структура, которая хранит все возможные пути стоимости s для данного агента. Каждая ее вершина соответствует возможному положению агента в момент t и находится на определенной глубине по отношению к стартовой вершине. Если существует кардинальный конфликт (a_i, a_j, v, t) , то ширины диаграмм для каждого агента и их путей оптимальной длины в момент времени t равны 1.

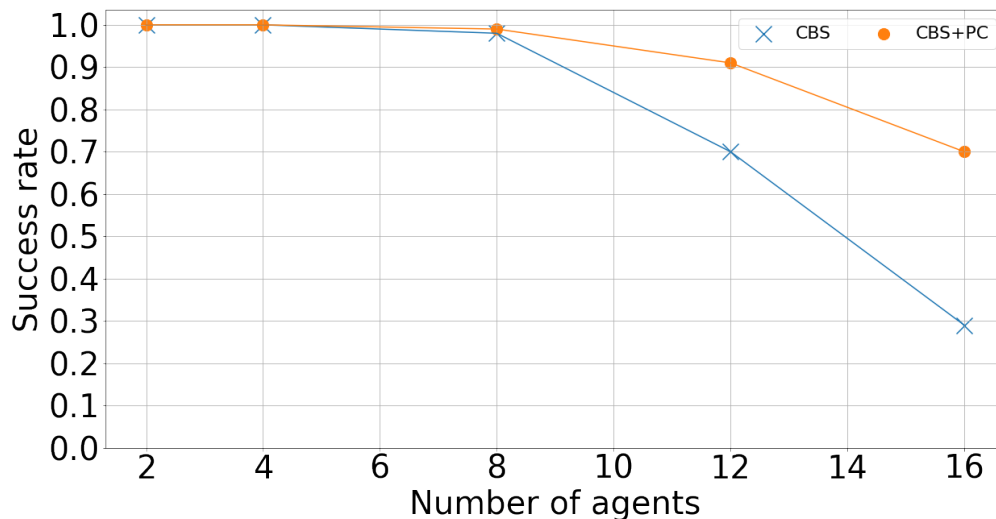
В данной работе используется упрощенный аналог MDD - ищутся все решения, стоимость которых равна первому найденному оптимальному. Для каждого из таких решений получается его путь из узлов $(v_1, v_2 \dots)$. Для каждого момента времени t создается множество из узлов. Для каждой найденной цепочки узлов узел добавляется в множество, соответствующее моменту времени. По размеру множества можно определить ширину.

Таким образом, для каждого текущего N конфликты проверяются на кардинальность (тестируется каждый конфликт (a_i, a_j, v, t) , то есть каждая пара агентов в конфликте). Если найден кардинальный конфликт, используется он. Если нет кардинальных конфликтов, берется первый найденный полукардинальный. В противном случае берется произвольный некардинальный конфликт.

Результаты экспериментов

В работе исследовалось движение агентов по двумерной карте; агентам позволено перемещаться в 4 направлениях: вверх, вниз, влево, вправо. В A* на нижнем уровне использовалось расстояние Манхэттена.

Сравнивались оба алгоритма: CBS и CBS с приоритизацией конфликтов.



На графике показана доля успехов (доля задач, решенных алгоритмом за 5 минут) для CBS и CBS+PC на 100 случайных картах 8x8 с 15% препятствий в зависимости от числа агентов (2, 4, 8, 12, 16; задачи для агентов также генерировались случайным образом). CBS+PC превосходит CBS по доле успехов.

	Время, с (CBS)	Время, с (CBS+PC)	CTNodes (CBS)	CTNodes (CBS+PC)
k				
2	0.001437	0.002621	1.900000	1.900000
4	0.007425	0.011000	6.800000	5.560000
8	3.085565	0.243648	2193.938776	93.612245
12	20.321756	1.345496	13922.614286	504.785714
16	79.022291	11.631259	69603.206897	3230.827586

В таблице представлены средние времена работы и числа сгенерированных вершин дерева ограничений для этих алгоритмов (усреднение шло по задачам, с которыми справились оба алгоритма) для разного числа агентов. Виден выигрыш CBS+PC как и в числе CTNodes, так и во времени работы. Таким образом, с помощью приоритизации конфликтов можно добиться повышения эффективности CBS.