ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 100 за 2019 г.

ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

Мельничук А.В., Сивакова Т.В., Судаков В.А.

Решение задач оптимизации с использованием мультиагентных моделей

Рекомендуемая форма библиографической ссылки: Мельничук А.В., Сивакова Т.В., Судаков В.А. Решение задач оптимизации с использованием мультиагентных моделей // Препринты ИПМ им. М.В.Келдыша. 2019. № 100. 16 с. doi:10.20948/prepr-2019-100
URL: http://library.keldysh.ru/preprint.asp?id=2019-100

О р д е н а Л е н и н а
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В.Келдыша
Р о с с и й с к о й а к а д е м и и н а у к

А.В. Мельничук, Т.В. Сивакова, В.А. Судаков

Решение задач оптимизации с использованием мультиагентных моделей

Москва — 2019

Мельничук А.В., Сивакова Т.В., Судаков В.А.

Решение задач оптимизации с использованием мультиагентного подхода

В работе исследуются подходы мультиагентного моделирования для решения оптимизационных задач с ограничениями. Рассмотрена задача построения сети взаимодействующих агентов. Каждый из агентов управляет определённым набором переменных и взаимодействует с агентами, имеющими с ним общие ограничения. Агенты функционируют как независимые параллельные процессы и обмениваются сообщениями с целью поиска оптимального значения целевой функции. Этот подход успешно применен к задаче составления расписаний.

Ключевые слова: мультиагентное моделирование, оптимизация с ограничениями, функция предпочтений агентов, распределенные системы

Alexander Vladimirovich Melnichuk, Tatyana Vladimirovna Sivakova, Vladimir Anatolievich Sudakov

Solving optimization problems using multi-agent models

In the study multi-agent modeling approaches researched for solving optimization problems with constraints. The problem of building a network of interacting agents is considered. Each of the agents manages a certain set of variables and interacts with agents that have common constraints with it. Agents function as independent parallel processes and exchange messages in order to find the optimal value of the objective function. This approach has been successfully applied to the task of scheduling.

Key words: multi-agent modeling, constrained optimization, agent preference function, distributed systems

Работа выполнена при поддержке Российского фонда фундаментальных исследований, проект № 18-00-00012 (18-00-00011) КОМФИ.

Введение

В данной работе рассматривается проблема распределения множества задач на множество ресурсов (таких как пакет заданий для нескольких процессоров, планирование лекций для аудиторий). Для решения столь важной и распространённой в последнее время проблемы предлагается использовать мультиагентный подход.

Один из примеров – проблема планирования собраний, которая актуальна для крупных организаций (например, государственных корпораций, частных компаний, исследовательских институтов и т.д.), когда многие люди, возможно работающие в разных отделах, участвуют в большом количестве рабочих встреч. Люди при планировании встреч зачастую имеют различные личные предпочтения в отношении времени начала встречи, например, конкретный сотрудник может предпочесть, чтобы его встречи начинались вечером, а не утром, поэтому приходится учитывать личные предпочтения участников, максимизируя их, для того чтобы выстроить расписание встреч. Необходимое требование к расписанию – соответствие очевидным жестким ограничениям, например, две встречи, в которых участвует один участник, не могут пересекаться.

Часто ограничения в обычной жизни воспринимаются как элементы, ограничивающие решения проблем, с которыми мы сталкиваемся (например, выбор, который мы делаем каждый день, обычно ограничен деньгами или временем). С вычислительной точки зрения, ограничения являются ключевыми компонентами для эффективного решения сложных задач. На практике ограничения кодируют знания о рассматриваемой проблеме и таким образом ограничивают пространство возможных решений, которые необходимо учитывать, значительно сокращая вычислительные затраты, необходимые для решения проблемы.

Поэтому обработку ограничений можно рассматривать как широкую и разнообразную область исследований, объединяющую методы и алгоритмы, которые охватывают многие различные дисциплины, включая исследование операций, компьютерное зрение, искусственный интеллект и теорию принятия решений. Все эти области имеют дело со сложными проблемами, которые можно сделать более понятными, тщательно продумав ограничения, определяющие структуру проблемы.

В данной работе показано, как обработка ограничений может реализовываться для решения проблем оптимизации в мультиагентных системах (MAS - multi-agent system), в первую очередь будут рассматриваться распределенные задачи оптимизации с ограничения (DCOP - distributed constraint optimization). В DCOP набор агентов должен прийти к какому-то соглашению (обычно через какую-то форму переговоров) о том, какие действия должен предпринять каждый агент, чтобы совместно получить наилучшее решение для всей системы [1]. Эта структура успешно используется не только

для планирования собраний, но и в сенсорных сетях, где датчики должны договориться о том, на какую цель они должны ориентироваться, чтобы получить наиболее точную оценку местоположения целей. Для тестирования эффективности алгоритмов решения DCOP часто используют задачи проверки выполнимости булевых функций и раскраски графов. Особенностью DCOP для MAS является то, что каждый агент ведет локальные переговоры только с подмножеством других агентов (обычно называемых соседями), которые могут непосредственно влиять на его поведение. В зависимости от постановки задачи и используемой методики решения это может значительно сократить процесс вычисления, с которыми сталкивается каждый агент, что делает сложные проблемы доступными даже для крупномасштабных систем. Например, обсуждавшийся ранее задаче планирования собраний, агент будет напрямую вести переговоры только с теми агентами, с которыми он должен встретиться, что обычно составляет небольшое подмножество агентов, вовлеченных во всю проблему.

Возможная формализация DCOP для задачи планирования собрания включает набор агентов, представляющих людей, участвующих в собрании, и набор переменных, которые представляют возможное время начала данного собрания в соответствии с участником. Ограничения предписывают равенство переменным, представляющим время начала одного и того же собрания разных агентов, и гарантируют, что переменные, которые представляют время начала разных встреч одного и того же агента, не совпадают. Наконец, предпочтения могут быть представлены как мягкие ограничения на время начала встреч, и общая цель состоит в том, чтобы оптимизировать сумму всех мягких ограничений. Хотя в этом параметре у нас есть личные предпочтения, мы максимизируем сумму предпочтений всех агентов, и, таким образом, рассматривается сценарий, когда агенты полностью сотрудничают, то есть они готовы уменьшить свою собственную локальную полезность, если это максимизирует глобальную полезность [2].

Несмотря на то что многие задачи можно решить, используя классическую оптимизацию, отличительная особенность предлагаемого распределенного подхода заключается в надежном и масштабируемом решении.

Определение сети ограничений

Ключевым элементом для распределенной обработки ограничений является концепция сети ограничений. Приведем стандартные формальные определения, относящиеся к сетям ограничений, а затем рассмотрим саму парадигму распределенной обработки ограничений.

Сеть ограничений N формально определяется как кортеж $<X, D, C>$, где $X = \{x_1, ..., x_n\}$ – набор дискретных переменных, $D = \{D_1, ..., D_n\}$ – это множество доменов переменных, которые перечисляют все возможные значения соответствующих переменных, а $C = \{C_1, \cdots, C_m\}$ представляет собой набор ограничений. Ограничение C_i может быть двух типов: жесткое или мягкое.

Жесткое ограничение C_i^h — это отношение R_i, определенное для подмножества переменных $S_i \subseteq X$. Переменные в S_i являются областью действия ограничения, а отношение R_i перечисляет все допустимые совместные комбинации значений переменных в области действия ограничения. Следовательно, R_i является подмножеством декартового произведения доменов переменных, которые находятся в пределах ограничения: $R_i \subseteq D_{i1} \times \cdots \times D_{ir}$, где $r = | S_i |$ — это арность отношения. Мягкое ограничение C_i^s — это функция F_i, определенная для подмножества переменных $S_i \subseteq X$, которые составляют область определения функции. Каждая функция F_i отображает каждую комбинацию значений всех переменных в области видимости на вещественное значение, то есть F_i: $D_{i1} \times \cdot \cdot \times D_{ir} \Rightarrow \Re$, где $r = | S_i |$ – арность функции.

Ограничения могут быть определены для любого подмножества переменных, однако наиболее наглядно работа в сетях ограничений (как алгоритмах решения, так и теоретическом анализе) показывается на бинарных сетях ограничений, где каждое ограничение (мягкое или жесткое) определяется двумя переменными.

Если набор ограничений включает только жесткие ограничения, то проблема называется проблемой удовлетворения ограничений (constraint satisfaction problem – CSP). В этом случае ищутся значения для всех переменных в сети, которые удовлетворяют всем ограничениям. Значения переменных удовлетворяют ограничению, если кортеж значений переменных принадлежит отношению ограничения, если $(a_{i1}, a_{i2}, \ldots, a_{ir}) \in R_i$, где $a_j \in D_j$ и $S_i = \{x_{i1}, x_{i2}, \ldots, x_{ir}\}$, такое назначение относится к решению для сети.

Если набор ограничений включает мягкие ограничения, то мы сталкиваемся с проблемой условной оптимизации (constraint optimization problem – COP) и наша цель – найти оптимальное решение. Это такая комбинация значений всех переменных, которая удовлетворяет всем ограничениям и которая максимизирует глобальную функцию $F(\overline{a})$. Глобальная функция $F(\overline{a})$ является суммой функций, представляющих мягкие ограничения: $F(\overline{a}) = \sum_i F_i(\overline{a}_i)$, где $\overline{a} = (a_1, a_2, ..., a_n)$, где $a_j \in D_j$, и $\overline{a}_i$ – это подмножество $\overline{a}$ для S_i. COP – может быть задачей максимизации или минимизации. Без ограничения общности далее сосредоточимся на задачах максимизации, поэтому наша цель – найти такое $\overline{a}^*$, которое удовлетворяет всем жестким ограничениям и такое, что:

$$\overline{a}^* = \arg\max_{\overline{a}} \sum_i F_i(\overline{a}_i) \tag{1}$$

В целом каждая задача CSP может также рассматриваться как задача COP, где мы стремимся найти назначение, которое нарушает наименьшее количество ограничений. Это особенно полезно для задач с большим количеством ограничений, где решение для CSP может не существовать. В частности, мы можем назначить постоянную фиксированную стоимость каждому

нарушенному ограничению и искать назначение, которое минимизирует сумму затрат. Такая задача называется Max-CSP.

Более того, всегда можно кодировать жесткие ограничения, используя только мягкие ограничения. При этом «штраф» на порядок больше любого допустимого значения целевой функции.

Например, предположим, без ограничения общности, что решается задача максимизации, и пусть R_i – отношение, соответствующее строгому ограничению C_i. Мы можем построить функцию $F_i(\overline{a}_i) = -\infty$, если $\overline{a}_i \notin R_i$, и $F_i(\overline{a}_i) = 0$ в остальных случаях. Следует отметить, что это может привести к ухудшению процедуры поиска решения, так как явные жесткие ограничения могут быть использованы для сокращения пространства поиска решения [3].

Распределенное решение задачи

Распределенное решение предполагает использование набора агентов, которые контролируют переменные и взаимодействуют, чтобы найти решение для сети ограничений. Как было сказано выше, это могут быть задачи CSP или COP, которые решаются соответствующими распределенными методами: распределенный CSP (distributed CSP – DCSP) и распределенный COP (distributed COP – DCOP). Парадигма DCSP была первоначально предложена для решения проблем координации в среде с несколькими агентами [4], однако в последние годы платформе DCOP уделялось больше внимания, поскольку она имеет больше сценариев практического применения и CSP можно свести к COP.

Формально DCOP может быть представлен в виде сети $N = \langle X, D, C \rangle$, содержащей мягкие ограничения, плюс набор агентов $A = \{A_1, A_2, \ldots, A_k\}$. Поиск оптимального решения DCOP – это NP-сложная проблема. Поэтому эмпирическая оценка методов решения DCOP является решающим моментом для оценки их возможного практического применения.

Точные алгоритмы решения

В данной работе рассмотрим точные методы решения, то есть те, которые всегда находят решение, которое соответствует наилучшему значению целевой функции (глобальный оптимум). Эти методы особенно интересны и изящны с теоретической точки зрения, но, поскольку мы имеем дело с проблемой NP-полноты, они также демонстрируют экспоненциально увеличивающиеся издержки координации либо в размере, либо в количестве сообщений, которыми обмениваются агенты, либо в вычислениях, проводимых каждым агентом.

В целом эти полные подходы можно разделить на два класса: те, которые основаны на поиске [5-9], и те, которые используют динамическое программирование [10]. Кроме того, подходы, основанные на поиске, делятся на синхронные, такие как SyncBB [9] и AND/OR поиск [8, 11], и асинхронные,

такие как ADOPT[5], NCBB[6] и AFB[7]. В модели синхронного выполнения агенты ждут сообщений от других агентов, прежде чем вычислять и отправлять новые сообщения сами. Напротив, в асинхронной модели выполнения агенты выполняют вычисления и отправляют сообщения, не ожидая сообщений от своих соседей. Асинхронная операция желательна в мультиагентном контексте, поскольку она позволяет агентам принимать решения, не дожидаясь, пока другие агенты завершат свои вычисления, полностью используя параллельные вычисления. С другой стороны, синхронная модель гарантирует, что агенты всегда обладаю самой актуальной информацией перед выполнением вычислений, таким образом сводя к минимуму избыточность как в вычислениях, так и в коммуникации.

Все вышеперечисленные методы полностью децентрализованы, в том смысле, что каждый агент имеет полный контроль над своими переменными и знает только про релевантные ограничения. Тем не менее, централизация части проблемы иногда может уменьшить усилия, необходимые для поиска оптимального в глобальном масштабе решения. Данная концепция лежит в основе подхода оптимального асинхронного частичного наложения (Optimal Asynchronous Partial Overlay – optAPO) [12]. Точнее, алгоритм optAPO стремится обнаружить части проблемы, которые особенно трудно решить децентрализованным способом (части, которые сильно взаимосвязаны), и централизует их в подзадачи, которые делегируются агентам-посредникам, которые действуют как централизованные решатели). Было показано, что optAPO последовательно снижает накладные расходы на связь по сравнению с другими децентрализованными методами, такими как ADOPT. Однако очень трудно предугадать, какая часть задачи будет решаться централизованно, и, следовательно, трудно предсказать затраты вычислительных ресурсов, которые потратят агенты-посредники.

Далее мы сосредоточимся на одном из децентрализованных подходов, который позволяет находить точное решение задач, представленных в общем виде.

Метод ADOPT

Асинхронная распределенная оптимизация (Asynchronous Distributed OPTimization – ADOPT) гарантирует оптимальность решения и позволяет агентам асинхронно изменять значения переменных, которыми они управляют [5, 13]. Оригинальная формулировка ADOPT предполагает, что каждый агент контролирует только одну переменную и ограничения накладываются на пары переменных, хотя существуют способы исключить эти предположения без существенного изменения алгоритма. Каноническое описание ADOPT предполагает, что решается задача минимизации. Ограничения представляют затраты, и агенты хотят найти назначение, которое минимизирует сумму этих затрат.

ADOPT – это метод, основанный на поиске, который выполняет распределенный поиск в обратном направлении с использованием стратегии «лучший первый»; каждый агент всегда присваивает своей переменной наилучшее значение на основе локальной информации. Ключевые компоненты алгоритма ADOPT:

- оценка локальной нижней границы,
- пороги возврата,
- условия завершения.

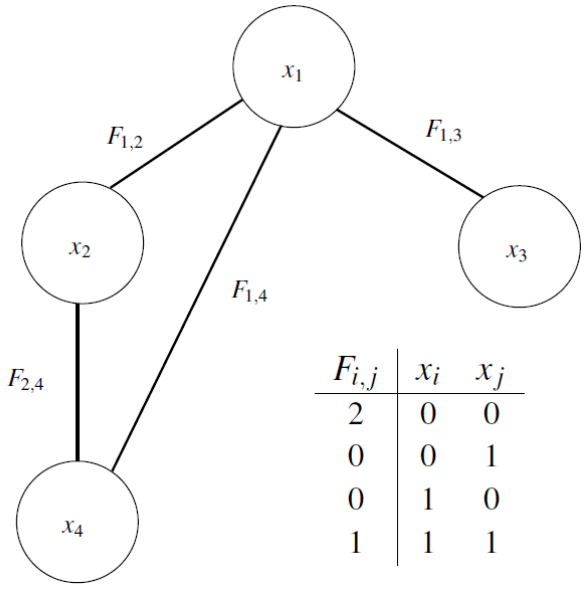
В частности, каждый агент поддерживает оценку нижней границы для каждого возможного значения его переменной. Эта нижняя граница первоначально рассчитывается на основе только локальной функции стоимости, а затем уточняется по мере передачи дополнительной информации между агентами. Каждый агент выберет значение своей переменной, которое минимизирует эту нижнюю границу, и это решение принимается асинхронно, как только обновляется локальная нижняя граница.

Пороги возврата используются для ускорения поиска ранее исследованных решений. Возврат может произойти, потому что стратегия поиска основана на локальных нижних границах, и, таким образом, агенты могут отказаться от значений, прежде чем для них будет проверена субоптимальность. Пороги возврата являются нижними границами, которые были определены ранее, и должны помешать агентам исследовать бесполезные ветви дерева поиска.

Наконец, ADOPT использует определенный ограниченный интервал для оценки прогресса поиска. Каждый агент поддерживает не только нижнюю границу, но и верхнюю границу оптимального решения. Следовательно, когда эти два значения совпадают, процесс поиска может быть прекращен, и текущее решение фиксируется. Кроме того, этот интервал может использоваться для поиска решений, которые являются неоптимальными, но находятся в пределах заданной предельной границы оптимального решения. Пользователь может указать допустимую границу ошибки (то есть расстояние между значением оптимального решения и приемлемым субоптимальным решением), и как только интервал привязки станет меньше этого значения, процесс поиска может быть остановлен.

Перед выполнением алгоритма ADOPT агенты должны быть упорядочены в порядке обхода дерева в глубину (depth first search - DFS). Порядок DFS определяет отношения предок-потомок между агентами. Упорядоченные деревья DFS часто используются при оптимизации [10], поскольку они имеют два интересных свойства:

- агенты в разных ветвях дерева не имеют общих ограничений,
- каждая сеть ограничений может быть упорядочена в дерево DFS, и это можно сделать за полиномиальное время с помощью распределенной процедуры [11].



$F_{i,j}$	x_i	x_j
2	0	0
0	0	1
0	1	0
1	1	1

Рис. 1. Пример сети ограничений

Тот факт, что агенты в разных ветвях не имеют общих ограничений, является важным свойством, гарантирует, что они могут искать решения независимо друг от друга. На рисунке 1 показана примерная сеть ограничений, а на рисунке 2 показан возможный порядок DFS, где сплошные линии показывают отношения предок-потомок, а ограничения не представлены. Упорядочение DFS может не являться уникальным, а производительность ADOPT (с точки зрения затрат на координацию) зависит от фактического используемого упорядочения DFS. Поиск оптимального дерева DFS – сложная проблема, которая в методе ADOPT не решается.

Учитывая порядок агентов в DFS, алгоритм продолжает обмениваться сообщениями трех типов: «Значение», «Стоимость» и «Порог». Когда алгоритм запускается, все агенты выбирают случайное значение для своих переменных, инициализируют нижнюю и верхнюю границу возможных значений стоимости. Эти границы затем уточняются по мере того, как по сети передается больше информации.

На рисунке 2 показаны сообщения, которыми обмениваются агенты на первых этапах алгоритма. Поскольку алгоритм является асинхронным, здесь указан один из возможных порядков выполнения, а числа в квадратах указывают частичный порядок сообщений.

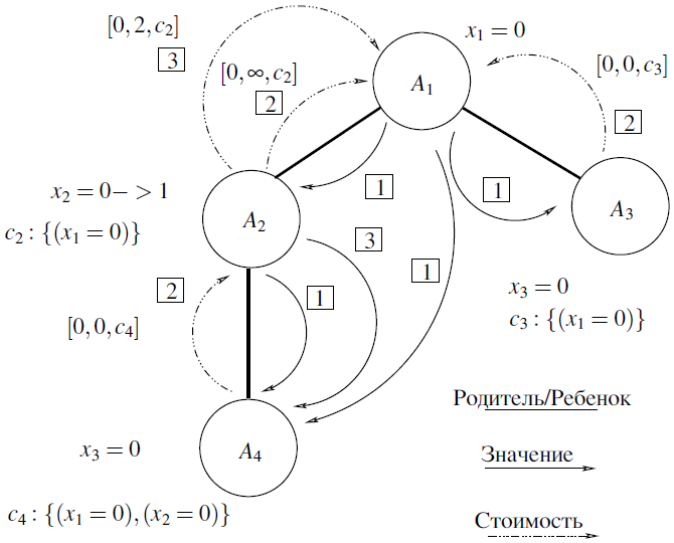


Рис. 2. Обмен сообщениями в алгоритме ADOPT: сообщения о значении и стоимости для одной возможной последовательности выполнения. Числа в квадратах указывают частичный порядок сообщений

Более подробно: сообщения «Значение» отправляются агентом всем его соседям, которые ниже в дереве DFS, чем он сам, сообщая о значении, которое агент присвоил своей переменной. Например, на рисунке 2 агент A_1 отправляет три сообщения «Значение» агентам A_2, A_3 и A_4, информируя их о том, что его текущее значение равно 0. Это сообщение отправляется в A_4, даже если между A_1 и A_4 нет отношений родитель-ребенок, поскольку A_4 является потомком A_1, который ниже в дереве DFS.

Сообщения «Стоимость» отправляются агентом его родителю, сообщая минимальную нижнюю и верхнюю границы для всех значений переменных агента и текущий контекст. Текущий контекст содержит значения всех вышестоящих предков. Например, на рисунке 2 текущий контекст A_4 — это c_4, он равен $\{(x_1, 0), (x_2, 0)\}$. Минимальная нижняя граница и минимальная верхняя граница вычисляются относительно текущего контекста. Для вычисления минимальной нижней границы каждый агент оценивает собственную локальную стоимость для каждого возможного значения своей переменной, добавляя стоимость из всех сообщений о нижней границе, полученных от дочерних агентов, которые связаны с текущим значением переменной. Локальная стоимость для агента является суммой значений функции локальной стоимости по всем предкам. Например, рассмотрим сообщение о стоимости, отправленное A_4. Минимальная нижняя граница вычисляется путем нахождения минимума между $\delta(x_4 = 0) = 4$ и $\delta(x_4 = 1)=0$, где $\delta(a)$ – функция локальной стоимости, когда переменная принимает значение a. Таким образом, минимальная нижняя граница A_4 в текущем контексте равна 0. Для агентов, не имеющих дочерних, верхняя граница равна нижней границе.

Сообщения о стоимости для агентов, которые не являются листами дерева DFS (например, A_2), также включают нижнюю и верхнюю границы для каждого дочернего элемента. Например, рассмотрим сообщение о стоимости, отправленное из A2 в A1, и пусть ***LB*** представляет минимальную нижнюю границу. ***LB*** вычисляется путем нахождения минимума между $\boldsymbol{LB}(x_2 = 0) = \delta(x_2 = 0) + lb(x_2 = 0, x_4) = 2$ и $LB(x_2 = 1) = \delta(x_2 = 1) + lb(x_2 = 1, x_4) = 0$, в результате чего $\boldsymbol{LB}$=0. Здесь $lb(a, x_l)$ – нижняя граница для дочерней переменной x_l, когда текущей переменной присваивается значение a в текущем контексте. Верхняя граница вычисляется аналогично, только вместо $lb(a, x_l)$ используется $ub(a, x_l)$ – верхняя граница для дочерней переменной x_l.

Сообщения «Порог» отправляются от родителей к детям, чтобы обновить пороги возврата агента. Пороги возврата – это особая переменная, которая используется при повторном анализе ранее рассмотренного контекста. Каждый агент сохраняет информацию о стоимости (верхнюю и нижнюю границы) только для текущего контекста и удаляет ранее сохраненную информацию, как только контекст изменяется. Фактически, если бы агент сохранял такую информацию для каждого посещенного контекста, ему понадобилось бы экспоненциальное пространство в памяти. Однако, поскольку контекст может посещаться несколько раз в процессе поиска, всякий раз, когда это происходит, агент должен вычислять информацию о стоимости с нуля. Но если этот контекст был посещен ранее, и агент сообщил информацию о стоимости своему родителю, так как родительский объект хранит эту информацию, он может отправить ее обратно агенту через пороговое сообщение. Сообщение о пороге используется для установки порога агента на нижнюю границу и распространения информации о стоимости вниз по дереву, чтобы избежать ненужных вычислений. Информация, которую хранит родитель, представляет собой накопленную информацию о стоимости. Поэтому, чтобы распространить информацию по дереву, агент должен разделить эту накопленную стоимость на свои дочерние элементы, используя некоторую эвристику, поскольку первоначальное разделение затрат потеряно, и исправить это разделение тогда, когда от детей будет получена обратная связь по стоимости. Например, предположим, что в процессе поиска x_1 меняет свое значение, а затем снова посещается контекст с $x_1 = 0$. Агент A_1 отправит пороговые сообщения в A_2 и A_3, как показано на рисунке 3.

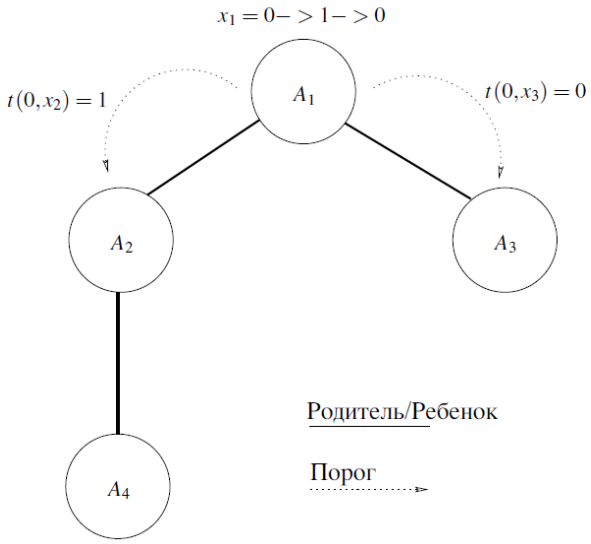


Рис. 3. Обмен сообщениями в алгоритме ADOPT: пороговые сообщения для повторного использования

Обратите внимание, что значением этих сообщений является значение нижней границы, отправленное соответствующим дочерним агентам, например, сообщение $t(x_1 = 0, x_2)$ равно нижней границе, отправленной A_2 в A_1 с контекстом $\{(x_1,0)\}$.

Агенты асинхронно обновляют значение переменной всякий раз, когда сохраненная нижняя граница для текущего значения превышает полученный от родителя порог и текущее значение переменной минимизирует нижнюю границу. Например, рассмотрим агента A_2 на рисунке 2 при получении сообщения от A_1. В этом случае нижняя граница для текущего значения ($\boldsymbol{LB}(x_2 = 0) = 2$) превышает пороговое значение (изначально установлено в 0). Поэтому агент обновляет значение своей переменной до значения, которое минимизирует нижнюю границу, которая в нашем случае равна $x_2 = 1$. Затем он отправляет сообщение о стоимости и значении. Если минимальная нижняя граница для значения переменной также является верхней границей для этого значения, агент может прекратить распространение сообщений, поскольку это значение будет оптимальным с учетом текущего контекста. Если для корневого агента это условие истинно, всем дочерним элементам отправляется сообщение о завершении. Агенты распространяют сообщение завершения, если условие завершения также верно для них. Когда сообщение о завершении распространяется на всех агентов, алгоритм останавливается, так как найдено оптимальное решение.

Апробация мультиагентных моделей проводилась с использованием пакета pyDCOP – это решатель DCOP написанный на языке Python [14]. Он обладает следующими особенностями:

- предоставляет реализации многих классических алгоритмов DCOP,

- позволяет легко реализовать собственные алгоритмы DCOP, предоставляя всю необходимую инфраструктуру: агенты, систему обмена сообщениями, сбор метрик,
- упрощает проведение распределенных экспериментов, так как агенты могут работать на одном и том же компьютере или на разных ЭВМ,
- обеспечивает мультиплатформенность, так как может работать на Windows, Mac и Linux,
- подходит для использования интернете вещей (IoT) и может запускать агенты на одноплатных компьютерах, таких как Raspberry Pi.

Задача формирования расписания была успешно решена на языке Python и размещена на портале веб-сервисов поддержки принятия решений ws-dss.com (см. рис. 4).

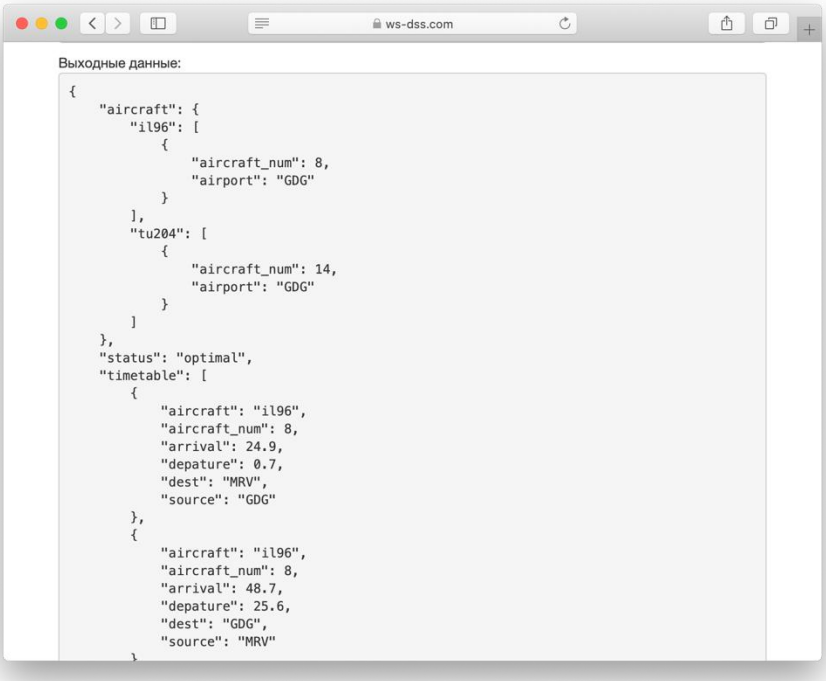


Рис. 4. Задача формирования расписания

Несколько работ, основанных на ADOPT, пытаются сократить время вычислений. Например, в работе [15] предложен метод BnB-ADOPT, который является расширением ADOPT, он последовательно сокращает время вычислений, используя различные стратегии поиска: поиск в глубину с методом ветвей и границ. В работе [16] предложено использование методов предварительной обработки для поиска ADOPT и показано, что это может привести к существенному увеличению производительности.

Заключение

В работе исследован эффективный мультиагентный метод – ADOPT, который работает в асинхронном режиме. Существенным преимуществом данного подхода по сравнению с динамическим программированием является экономное использование памяти каждым агентом, оно полиномиально по количеству переменных. Все сообщения имеют фиксированный размер, при этом количество сообщений, которыми должны обмениваться агенты, в худшем случае экспоненциально по количеству переменных. Это влияет на время, необходимое для поиска оптимального решения. В частности, число циклов синхронизации сообщений определяется числом всех агентов, получивших входящие сообщения и отправивших исходящие сообщения, и является экспоненциальным по количеству переменных. Число сообщений — это часто используемая мера для оценки методов решения DCOP, поскольку она менее чувствительна к изменениям скорости вычислений агентов и задержкам связи. Масштабируемость подхода могут серьёзно ограничивать экспоненциальные элементы, которые неизбежны при поиски точного оптимального решения.

MAS позволяет минимизировать количество информации, которую агенты должны раскрывать друг другу (таким образом, повышается уровень конфиденциальности), так как в DCOP агентам необходимо знать только об ограничениях, в которые они вовлечены.

В целом структура DCOP и алгоритмы взаимодействия агентов, разрабатываемые для решения трудноразрешимых задач, представляют собой актуальную область исследований в сообществе MAS, которая все чаще применяется в реальных условиях.

Библиографический список

1. A. Farinelli, M. Vinyals, A. Rogers, and N. Jennings, "Distributed Constraint Handling and Optimization", in G. Weiss (ed.), "Multiagent Systems" (second edition), MIT Press, p. 547-584, 2013.
2. Т.В. Сивакова, В.А. Судаков Метод нечетких областей предпочтения для оценки эффективности инноваций // XXVIII Международная научно-техническая конференция "Современные технологии в задачах управления, автоматики и обработки информации". Алушта, 14-20 сентября 2019 г. Сборник трудов. М.: Изд.-во Национальный исследовательский ядерный университет "МИФИ", 2019. С. 81-82.
3. R. Dechter. Constraint Processing. Morgan Kaufmann, 2003.
4. Makoto Yokoo. Distributed constraint satisfaction: Foundations of cooperation in multi-agent systems. Springer-Verlag, 2001.
5. P. J. Modi, W. Shen, M. Tambe, and M. Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. Artificial Intelligence Journal, (161):149–180, 2005.

6. A. Chechetka and K. Sycara. No-commitment branch and bound search for distributed constraint optimization. In Proceedings of Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, pages 1427–1429, 2006.
7. A. Gershman, A. Meisels, and R. Zivan. Asynchronous forward bounding for dis- tributed COPs. Journal Artificial Intelligence Research, 34:61–88, 2009.
8. R. Dechter and R. Mateescu. And/or search spaces for graphical models. Artificial Intelligence, 171:73–106, 2007.
9. Katsutoshi Hirayama and Makoto Yokoo. Distributed partial constraint satisfaction problem. In Principles and Practice of Constraint Programming, pages 222–236, 1997.
10. A. Petcu and B. Faltings. DPOP: A scalable method for multiagent constraint optimization. In Proceedings of the Nineteenth International Joint Conference on Arti- ficial Intelligence, pages 266–271, 2005.
11. A. Petcu. A Class of Algorithms for Distributed Constraint Optimization. PhD thesis, Swiss Federal Institute of Technology (EPFL), Lausanne (Switzerland), 2007.
12. R. Maillerand, V. Lesser. Solving distributed constraint optimization problems using cooperative mediation. In Proceedings of Third International Joint Conference on Autonomous Agents and MultiAgent Systems, pages 438–445, 2004.
13. P. J. Modi. Distributed constraint optimization for multiagent systems. PhD thesis, Department of Computer Science, University of Southern California, 2003.
14. https://github.com/Orange-OpenSource/pyDcop
15. W. Yeoh, A. Felner, and S. Koenig. BnB-ADOPT: An asynchronous branch- and- bound DCOP algorithm. In Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems, pages 591–598, 2008.
16. S. M. Ali, S. Koenig, and M. Tambe. Preprocessing techniques for accelerating the DCOP algorithm ADOPT. In Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, pages 1041–1048, 2005.

Оглавление

Введение 3
Определение сети ограничений 4
Распределенное решение задачи 6
Точные алгоритмы решения 6
Метод ADOPT 7
Заключение 14
Библиографический список 14