СОВРЕМЕННЫЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

УДК 004.89, 004.832.2 ISSN 1995-5499
DOI: https://doi.org/10.17308/sait.2020.3/3041
Поступила в редакцию 02.06.2020
Подписана в печать 30.09.2020

ГИБРИДНЫЕ МЕТОДЫ РЕШЕНИЯ ЗАДАЧ УДОВЛЕТВОРЕНИЯ НЕЧИСЛОВЫХ ОГРАНИЧЕНИЙ С ИСПОЛЬЗОВАНИЕМ СПЕЦИАЛИЗИРОВАННЫХ ТАБЛИЦ

© 2020 А. А. Зуенко✉

Институт информатики и математического моделирования
«Кольский научный центр Российской академии наук»
ул. Ферсмана, 24А, 184209 Апатиты, Мурманская обл., Российская Федерация

Аннотация. Для повышения эффективности представления и обработки качественных зависимостей предметной области в рамках технологии программирования в ограничениях, в статье предлагается использовать специализированные табличные структуры: *C*- и *D*-системы. С точки зрения технологии программирования в ограничениях данные структуры могут быть отнесены к такому классу глобальных ограничений как compressed таблицы. Впервые разработаны гибридные методы решения задач удовлетворения нечисловых ограничений, формализованных с помощью предлагаемых *C*- и *D*-систем. Отличительной чертой авторских методов является то, что они не используют стратегий поиска в глубину с возвратами, приводящих к экспоненциальному росту времени выполнения вычислительных процедур при росте размерности задачи. Первый из предложенных гибридных методов интегрирует авторские методы распространения нечисловых ограничений с существующими алгоритмами структурной декомпозиции графа ограничений. В результате декомпозиции задача разбивается на подзадачи, граф ограничений каждой из которых представляет собой дерево. В этом случае подзадачи решаются за полиномиальное время с помощью авторских алгоритмов распространения нечисловых ограничений. Метод хорошо подходит для ситуации, когда имеется серия задач с одной и той же структурой графа ограничений, например при анализе типовых запросов к хранилищам данных. Второй из разработанных гибридных методов интегрирует следующие компоненты: авторские методы распространения нечисловых ограничений для сокращения размерности пространства поиска; метод локального поиска на частичных присваиваниях для быстрого выхода из подпространств, не содержащих решение; поиск с запретами (tabu search) для избегания повторного прохождения уже исследованных состояний. Он предназначен для ситуаций, когда требуется найти решение на основе больших объемом информации, но при этом отсутствуют априорные сведения о структуре графа задачи удовлетворения ограничений.

Ключевые слова: задача удовлетворения ограничений, локальный поиск, распространение ограничений, структурная декомпозиция.

✉ Зуенко Александр Анатольевич
e-mail: zuenko@iimm.ru

ВВЕДЕНИЕ

В рамках программирования в ограничениях общепринятая методология поиска заключается в совместном применении методов распространения ограничений с методами поиска в глубину с возвратами, при этом применяются специализированные эвристики для выбора переменной и её значения, а также используются разумные стратегии возврата к состоянию, являющемуся причиной возникновения недопустимого присваивания. Особенность методов поиска с возвратами заключается в том, чтобы пошагово расширять частичное допустимое решение до полного. Если частичное решение недопустимо, то осуществляется возврат и предыдущее частичное решение расширяется в альтернативном направлении.

Во многих практических приложениях, связанных с обработкой большого объема информации, методы, опирающиеся только на исследование дерева поиска, оказываются недостаточно эффективными, и на передний план выходят методы локального поиска, а также гибридные методы, сочетающие преимущества нескольких схем поиска.

Важным типом так называемых глобальных ограничений в рамках технологии программирования в ограничениях являются табличные ограничения. В табличной форме удобно описывать различные нечисловые (качественные) зависимости предметной области. Однако, с ростом размера таблиц экспоненциально растет и сложность процедур их обработки. В связи с этим, именно для табличных ограничений актуальной представляется разработка способов их компактного представления, а также эффективных методов их обработки.

В зарубежной литературе тематике компактного представления табличных ограничений посвящен широкий спектр работ (см., например, [1–4]). Часть из них сконцентрирована на алгоритмах удовлетворения ограничений, представленных с помощью compressed-таблиц [1, 3, 4]. Также развивается подход для обработки таблиц в виде такого глобального ограничения как mddc, которое представляет задачу удовлетворения ограничений с помощью MDD-диаграмм (multi-valued decision diagram) [2]. Однако, упомянутые исследования посвящены преимущественно разработке алгоритмов распространения ограничений, которые обеспечивают вынуждение свойства GAC (Generalized Arc Consistency) для рассматриваемой задачи удовлетворения ограничений и предназначены для использования совместно с алгоритмами поиска в глубину с возвратами. Кроме того, исследования показали, что существующие типы табличных ограничений предназначены, прежде всего, для моделирования дизъюнктивных нормальных форм конечных предикатов и плохо подходят для моделирования конъюнктивных нормальных форм логических формул. При табличном представлении и анализе конъюнктивных нормальных форм задача вынуждения такого свойства как GAC может оказаться слишком затратной с точки зрения вычислительных ресурсов.

Настоящую работу можно рассматривать как продолжение серии авторских публикаций [5–8], посвященных исследованию свойств и разработке методов рассуждений на compressed-таблицах. В данных статьях, как и в настоящей работе, для повышения эффективности представления и обработки табличных ограничений предлагается использовать два вида compressed-таблиц: *C*- и *D*-системы. *D*-системы, по-сути, являются совершенно новым типом compressed-таблиц и служат для удобства моделирования продукционных правил и некоторых видов логических выражений (конъюнктивных нормальных форм конечных предикатов). Определения понятий *C*- и *D*- система а также начальные исследования их свойств, приводятся в [9], но там данные структуры не рассматриваются в контексте решения задач удовлетворения ограничений. Автором были разработаны специализированные правила редукции, которые сохраняют множества решений, но позволяют сократить размеры *D*-системы, а также методы поиска в глубину с возвратами для случая, когда ограничения представлены в виде *D*-систем [5, 7]. Кроме того, были систематизированы и изучены свойств *C*-си-

стем, сформулированы и доказаны теоремы, лежащие в основе правил редукции для случая *C*-систем [6].

Однако, оказалось, что в некоторых прикладных задачах методы систематического поиска в глубину с возвратами, даже с учетом предлагаемого «сжатого» представления табличных ограничений, не позволяют найти решение. Тем не менее, работ, где бы рассматривалось применение методов несистематического поиска для организации рассуждений с использованием табличных ограничений автору обнаружить не удалось. В [8] была предложена обобщенная схема организации локального поиска на ограничениях, представленных в форме *D*-систем.

В настоящей работе впервые для организации эффективного поиска при большой размерности предложенных видов compressed-таблиц разработаны гибридные методы, которые интегрируют методы локального поиска, распространения нечисловых ограничений, систематического поиска, а также известные методы структурной декомпозиции графа ограничений. Разрабатываемые методы опираются на детальный анализ внутренней структуры предлагаемых видов табличных ограничений.

Гибридные методы, разработанные на данный момент в рамках предлагаемого подхода, относятся к следующим двум классам: 1) методы, реализующие структурную декомпозицию графа ограничений совместно с распространением ограничений; 2) методы, реализующие распространение ограничений совместно с процедурами локального поиска.

1. ТАБЛИЧНЫЕ ОГРАНИЧЕНИЯ ДЛЯ ПРЕДСТАВЛЕНИЯ И ОБРАБОТКИ КАЧЕСТВЕННЫХ ЗАВИСИМОСТЕЙ

Чтобы прикладную задачу можно было решить в рамках технологии программирования в ограничениях, она должна быть представлена как некоторая задача удовлетворения ограничений.

Согласно [10] *задача удовлетворения ограничений* (Constraint Satisfaction Problem — CSP) состоит из трех компонент: $\langle X, D, C \rangle$: X — множество переменных $\{X_1, X_2, ..., X_n\}$, D — множество доменов $\{D_1, D_2, ..., D_n\}$, где D_i является областью определения переменной X_i, C — множество ограничений $\{C_1, C_2, ..., C_m\}$, которые предписывают допустимые комбинации значений переменных. Каждый домен D_i описывает множество допустимых значений $\{v_1, ..., v_k\}$ для переменной X_i. Каждое ограничение есть пара $\langle scope, rel \rangle$, где $scope$ — множество переменных, которые участвуют в ограничении, а rel — отношение, регламентирующее допустимые комбинации значений, которые переменные из $scope$ могут принимать.

Присваивание, которое не нарушает никаких ограничений, называется *допустимым* присваиванием. *Полным* называется такое присваивание, в котором участвует каждая переменная. *Решением задачи* CSP является полное присваивание, которое удовлетворяет всем ограничениям.

Ограничения могут быть представлены путем явного перечисления всех допустимых комбинаций значений для указанного набора переменных (экстенсиональный способ задания), а могут — в форме абстрактного отношения, поддерживающего две операции: проверка на принадлежность некоторого кортежа данному отношению, перечисления всех элементов отношения (интенсиональный способ задания). Примером экстенсиональных отношений могут служить таблицы в реляционных базах данных. Эти отношения играют важную роль в области САПР (систем автоматизированного проектирования) и в области ГИС (геоинформационных систем). Когда мы говорим об интенсиональных отношениях, то имеем в виду отношения, заданные параметрически. Таковыми являются почти все отношения над числовыми (непрерывными) областями.

Зачастую, многоместные отношения, заданные экстенсионально, могут быть выражены более компактно, чем полным перечислением своих кортежей.

Предлагаемый вниманию математический аппарат для «сжатого» представления многоместных отношений используют два типа матрицеподобных структур. Первый тип — это

C-системы. *C*-система — это compressed-таблица, где в качестве ячеек выступают не отдельные элементы, а множества.

Рассмотрим пример отношения: $\{(c,1), (c,2),(c,4), (c,5), (b,2), (b,4), (d,1), (d,5)\}$, которое задано в пространстве атрибутов X,Y с доменами: X — $\{a,b,c,d\}$, Y — $\{1,2,3,4,5\}$.

Ниже представлена *C*-система, которая компактно описывает представленное отношение, и алгебраическое выражение над множествами, которое соответствует данной *C*-системе:

$$T[XY]=\begin{bmatrix}\{c\} & \{1,2,4,5\}\\ \{b\} & \{2,4\}\\ \{d\} & \{1,5\}\end{bmatrix}=$$

$$=\{c\}\times\{1,2,4,5\}\cup\{b\}\times\{2,4\}\cup\{d\}\times\{1,5\}.$$

Графически каждая строка *C*-системы соответствует некоторой области в признаковом пространстве (декартову произведению множеств), а вся *C*-система — объединению этих областей.

Другой тип compressed-таблиц, обеспечивающий компактное представление многоместных отношений, — это *D*-система. Данная матрица записывается в обратных скобках. Ниже приводится *D*-система $P[X,Y]$, которая эквивалентна рассмотренной ранее *C*-системе $T[X,Y]$, поскольку обе эти структуры описывают одно и то же исходное отношение:

$$P[XY]=\begin{bmatrix}\{c,d\} & \{2,4\}\\ \{b,c\} & \{1,5\}\\ \varnothing & \{1,2,4,5\}\end{bmatrix}=$$

$$=(\{c,d\}\times\{1,2,3,4,5\}\cup\{a,b,c,d\}\times\{2,4\})\cap$$
$$\cap(\{b,c\}\times\{1,2,3,4,5\}\cup\{a,b,c,d\}\times\{1,5\})\cap$$
$$\cap(\{a,b,c,d\}\times\{1,2,4,5\}).$$

D-система представляет собой сложное алгебраическое выражение. Каждая строка *D*-системы представляет собой некоторую область в пространстве признаков, более сложно устроенную, чем декартово произведение множеств. Каждая строка *D*-системы — это объединение определенных декартовых произведений. Вся *D*-система описывает пересечение областей, соотнесенных с каждой из её строк.

Пустая компонента («∅» в описании *C*- и *D*-систем) — это фиктивная компонента, не содержащая значений. Другой фиктивной компонентой является полная компонента «*» — сокращенное обозначение всего диапазона возможных значений (домена) атрибута.

Как и к обычным таблицам, к *C*- и *D*-системам могут быть применены операции реляционной алгебры. Для случая *C*- и *D*-систем, определение таких базовых операций, как пересечение, соединение отношений и т. п. не отличается от определений, принятых в реляционной алгебре. Их выполнение осуществляется с использованием специализированных теорем без разложения *C*- и *D*-систем в обычные таблицы. Кроме того, к *C*- и *D*-системам широко применяется операция дополнения многоместных отношений, которая практически не используется в реляционных СУБД. Более подробно об операциях с *C*- и *D*-системами можно найти в [9]. Далее приведем теорему, которая потребуется для дальнейшего изложения [9]. Здесь, ввиду ограничений на объем статьи, она приводится без доказательств.

Теорема 1. *Результат соединения двух C-систем можно представить в виде C-системы, состоящей из строк, получаемых соединением каждой вектор-строки первой C-системы с каждой вектор-строкой второй C-системы.*

Далее перейдем к описанию разработанных гибридных методов поиска на предлагаемых видах «compressed» — таблиц.

2. СОВМЕСТНОЕ ПРИМЕНЕНИЕ МЕТОДОВ СТРУКТУРНОЙ ДЕКОМПОЗИЦИИ ГРАФА ОГРАНИЧЕНИЙ И АВТОРСКИХ МЕТОДОВ РАСПРОСТРАНЕНИЯ НЕЧИСЛОВЫХ ОГРАНИЧЕНИЙ

Разработан гибридный метод, который интегрирует авторские методы распространения нечисловых ограничений [6] с существующими методами структурной декомпозиции графа ограничений [11, 12]. Известно,

что если граф задачи (подзадачи) CSP имеет древовидную структуру, то для нахождения её решения достаточно применения только методов распространения ограничений. Особенностью разработанного метода является то, что множество решений каждой подзадачи ищется в виде некоторой *C*-системы, то есть за шаг поиска ищутся целые подпространства в пространстве допустимых присваиваний, что существенно сокращает время исполнения вычислительных процедур. Метод состоит из трех основных этапов:

1. Разбиение графа ограничений исходной задачи на слабосвязанные или независимые подзадачи, графы ограничений которых являются деревьями. На этом этапе применяются известные методы структурной декомпозиции графа ограничений (например, методы tree decomposition или методы cycle cutset).

2. Решение подзадач путем применения известного алгоритма решения задач CSP с древовидной структурой и авторских методов распространения нечисловых ограничений. Решение каждой из подзадач ищется в виде *C*-системы.

3. Соединение решений подзадач в решение общей задачи осуществляется с использованием Теоремы 1 и представляет собой итоговую *C*-систему, компактно описывающую множество решений.

Представление множества решений в форме *C*-систем существенно экономит ресурсы памяти компьютера и ускоряет вычислительные процедуры поиска решений по сравнению с явным перечислением всех возможных решений. В то же время не составляет трудности явно перечислить все элементарные решения на основе их представления в форме *C*-системы, поскольку *C*-система обладает простой характеристической функцией.

Итак, разработанный метод, интегрирует две основные компоненты: a) компоненту, реализующую алгоритмы распространения; b) компоненту, реализующую структурную декомпозицию исходной задачи. Далее рассмотрим принципы функционирования этих двух компонент.

Структурная декомпозиция графа ограничений

Рассмотрим основные подходы, которые лежат в основе структурных алгоритмов. Другими словами, опишем способы, позволяющие использовать для быстрого поиска решений структуру самой задачи, представленную в виде графа ограничений.

Структура или топология задач CSP может описываться с помощью различных графовых структур: (первичного) графа ограничений, гиперграфа ограничений, двойственного графа ограничений.

Первичный граф ограничений CSP (V, D, C) — это неориентированный граф $G = (V, E)$, вершины V которого соответствуют переменным CSP, причем две вершины соединяются ребром в графе G, если соответствующие переменные участвуют в одном и том же ограничении. Далее под графом ограничений будем понимать первичный граф. Граф ограничений может быть использован для разбиения всей задачи на совокупность более простых с точки зрения вычислительной сложности подзадач. Если в составе задачи CSP можно выделить полностью независимые друг от друга подзадачи, то их можно решать отдельно друг от друга, а полученные решения впоследствии скомбинировать в решение исходной задачи.

Для разбиения задачи CSP на независимые подзадачи $\{CSP_i\}$, можно рассмотреть связные компоненты графа ограничений. Предположим, что каждая подзадача CSP_i содержит c переменных из общего количества n переменных, где c — константа. В таком случае получится n/c подзадач, и для решения каждой из них потребуется, самое большее, объем работы d^c, где d — размер доменов переменных. Таким образом, общий объем работы измеряется величиной $O(d^c \cdot n/c)$, которая линейно зависит от n. Без такой декомпозиции общий объем работы измерялся бы величиной $O(d^n)$, которая экспоненциально зависит от n. Полностью независимые подзадачи являются очень привлекательными, но встречаются редко.

В большинстве случаев подзадачи любой задачи CSP связаны друг с другом по пере-

менным. В простейшем случае граф ограничений является деревом: любые две переменные связаны не больше чем одним путем. Алгоритм решения CSP с древовидной структурой обладает низкой вычислительной сложностью (задача CSP может быть решена за линейное время) [11].

Поскольку существует эффективный алгоритм для деревьев, следует рассмотреть вопрос о том, можно ли каким-то образом приводить к древовидным структурам более общие графы ограничений.

Существуют два основных способа решения этой задачи; один из них основан на удалении вершин, а другой — на слиянии вершин друг с другом и образовании супер-вершин. Первый подход предусматривает присваивание значений некоторым переменным так, чтобы оставшиеся переменные образовывали дерево. В рамках первого подхода наиболее известными методами являются метод множеств разрыва цикла [13] и метод нахождения связных компонент графа ограничений [14, 15]. К типичным методам, реализованным в рамках второго подхода являются: метод древовидной декомпозиции (tree decomposition) [12, 14, 16] и метод древовидной кластеризации (tree-clustering) [12].

Описанные выше методы структурной декомпозиции имеют следующие общие черты. Каждая подзадача общей задачи CSP решается независимо; если какая-либо из них не имеет решения, то вся задача также не имеет решения. Если удается решить все подзадачи, то предпринимается попытка составить глобальное решение.

Правила распространения ограничений, представленных в виде С-систем

Приведем утверждения, позволяющие реализовывать эквивалентные преобразования совокупности ограничений для случая, когда ограничения представлены в виде набора *С*-систем [6]. Целью преобразований является приведение CSP к более простому виду, где содержится меньшее количество *С*-систем, строк *С*-систем, столбцов (атрибутов) *С*-систем, значений в доменах атрибутов и т. п.

Утверждение 1 (*У1*). Если все строки (кортежи) *С*-системы пусты, то есть содержат хотя бы по одной пустой компоненте каждая, то *С*-система пуста (соответствующая задача CSP несовместна).

Утверждение 2 (*У2*). Если все компоненты некоторого атрибута (столбца *С*-системы) являются полными, то данный атрибут можно удалить из *С*-системы (удаляются все компоненты стоящие в соответствующем столбце), а пара «удаляемый атрибут — его домен» сохраняется в векторе частичного решения.

Утверждение 3 (*У3*). Если домен некоторого атрибута *С*-системы содержит значения, не встречающиеся в соответствующем столбце, то эти значения удаляются из данного домена.

Утверждение 4 (*У4*). Если строка *С*-системы содержит хотя бы одну пустую компоненту (строка пуста), то строка удаляется.

Утверждение 5 (*У5*). Если компонента некоторого атрибута содержит значение, не принадлежащее соответствующему домену, то это значение удаляется из компоненты.

Утверждение 6 (*У6*). Если одна строка *С*-системы полностью доминирует (покомпонентно содержит) другую строку, то доминируемая строка удаляется из *С*-системы.

Часть из приведенных утверждений позволяет исключать значения из доменов и компонент атрибутов (***У3***, ***У5***) или даже сами столбцы-атрибуты (***У2***), а часть дает возможность исключать из рассмотрения лишние строки (***У4***, ***У6***). *Признак успешного завершения* процесса поиска — элиминация из *С*-системы *всех* строк и столбцов без образования пустых строк. Другими словами, результирующее состояние в этом случае будет характеризоваться только совокупностью непустых усеченных доменов в векторе решения. *Признаком несовместности CSP* является пустота *С*-системы (***У1***).

Пример реализации предлагаемого гибридного метода

Пример 1. Пусть заданы следующие соответствия «атрибут — домен атрибута»:
$X \ — \ \{a_1, a_2, a_3, a_4, a_5, a_6\}$,

Y — $\{b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9\}$,
Z — $\{c_1, c_2, c_3, c_4, c_5, c_6\}$,
W — $\{d_1, d_2, d_3, d_4, d_5, d_6\}$,
L — $\{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}$,
M — $\{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_{10}, f_{11}, f_{12}, f_{13}, f_{14}\}$.

А также задана следующая совокупность ограничений, заданных в форме *C*-систем:

Ограничение $R_1[XY]$

$$\begin{bmatrix} \{a_1\} & \{b_1, b_2\} \\ \{a_2\} & \{b_3, b_4, b_5\} \\ \{a_3\} & \{b_6\} \\ \{a_4\} & \{b_7\} \\ \{a_5\} & \{b_8\} \\ \{a_6\} & \{b_9\} \end{bmatrix}$$

Ограничение $R_2[YZ]$

$$\begin{bmatrix} \{b_1\} & \{c_1\} \\ \{b_2, b_3, b_6\} & \{c_2\} \\ \{b_4\} & \{c_3\} \\ \{b_5, b_7\} & \{c_4\} \\ \{b_8\} & \{c_5\} \\ \{b_9\} & \{c_6\} \end{bmatrix}$$

Ограничение $R_3[WZ]$

$$\begin{bmatrix} \{d_1\} & \{c_1\} \\ \{d_2\} & \{c_2\} \\ \{d_3\} & \{c_3\} \\ \{d_4\} & \{c_4\} \\ \{d_5\} & \{c_5\} \\ \{d_6\} & \{c_6\} \end{bmatrix}$$

Ограничение $R_4[LW]$

$$\begin{bmatrix} \{e_1\} & \{d_1\} \\ \{e_2\} & \{d_2, d_3\} \\ \{e_3\} & \{d_2\} \\ \{e_6\} & \{d_4\} \\ \{e_7\} & \{d_5, d_6\} \end{bmatrix}$$

Ограничение $R_5[ZM]$

$$\begin{bmatrix} \{c_1\} & \{f_1, f_2\} \\ \{c_2\} & \{f_3, f_4, f_5\} \\ \{c_3\} & \{f_6, f_7\} \\ \{c_4\} & \{f_8, f_9, f_{10}\} \\ \{c_5\} & \{f_{11}, f_{12}\} \\ \{c_6\} & \{f_{13}, f_{14}\} \end{bmatrix}$$

Ограничение $R_6[ML]$

$$\begin{bmatrix} \{f_1, f_6\} & \{e_1\} \\ \{f_2, f_3, f_4\} & \{e_2\} \\ \{f_4, f_6\} & \{e_3\} \\ \{f_5, f_7\} & \{e_4\} \\ \{f_8, f_9, f_{11}\} & \{e_5\} \\ \{f_9, f_{11}, f_{14}\} & \{e_6\} \\ \{f_{10}, f_{13}\} & \{e_7\} \\ \{f_{12}, f_{14}\} & \{e_8\} \end{bmatrix}$$

Задача состоит в том, что требуется вычислить соединение данных отношений, то есть требуется найти: $R_1[XY] \oplus R_2[YZ] \oplus \oplus R_3[WZ] \oplus R_4[LW] \oplus R_5[ZM] \oplus R_6[ML]$.

На рис. 1 представлен граф задачи CSP:

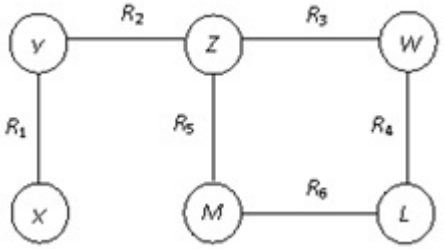

Рис. 1. Граф исходной задачи CSP
[Fig. 1. Graph of the initial CSP]

На рис. 2 приведен пример структурной декомпозиции графа исходной задачи CSP, который рассмотрен нами ранее (рис. 1). Разбиение исходной задачи CSP на подзадачи выполняется путем присваивания значений переменной Z.

Для каждой подзадачи CSP, полученной в процессе структурной декомпозиции, можно заранее вычислить соответствующее многоместное отношение. Затем, эти отношения можно соединить в одно.

В этом случае, для решения исходной задачи требуется рассмотреть две подзадачи CSP:

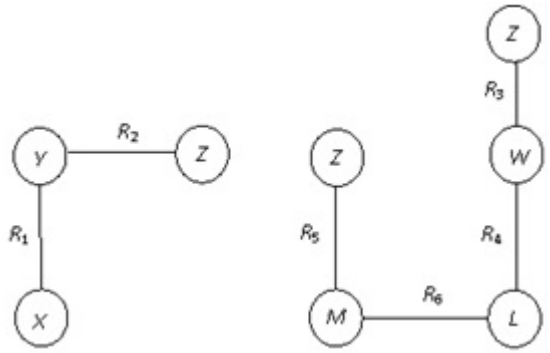

Рис. 2. Эффект структурной декомпозиции после присваивания значений переменной Z
[Fig. 2. The effect of structural decomposition after assigning values to the variable Z]

1) $R_2[YZ] \oplus R_1[XY]$;

2) $R_3[WZ] \oplus R_4[LW] \oplus R_6[ML] \oplus R_5[ZM]$.

Предположим, что переменной Z присвоено значение c_1, тогда домен переменной Z становится одноэлементным множеством $\{c_1\}$.

Рассмотрим первую подзадачу CSP $(R_2[YZ] \oplus R_1[XY])$. Граф данной подзадачи является деревом, следовательно, решение подзадачи можно получить, используя лишь разработанные ранее методы распространения нечисловых ограничений без привлечения методов backtracking. Сначала активируется ограничение $R_2[YZ]$ и производится «настройка» данного ограничения на новый одноэлементный домен переменной Z на основании утверждений из списка ***У1***-***У5***, приведенного выше. После применения утверждения ***У5*** получаем, что все строки исходной *C*-системы $R_2[YZ]$, кроме первой, становятся пустыми. Тогда, согласно ***У4***, ограничение $R_2[YZ]$ примет вид: $[\{b_1\}\{c_1\}]$. Теперь можно заключить, применяя ***У3***, что новым доменом переменной Y будет синглетон $\{b_1\}$. Далее активируется ограничение $R_1[XY]$, которое после «настройки» на новый домен переменной Y принимает вид: $[\{a_1\}\{b_1\}]$. Итак, мы закончили движение в направлении от ограничения $R_2[YZ]$ к ограничению $R_1[XY]$.

Движение в противоположном направлении можно представить как результат соединения «усеченных» ограничений. Имеем следующую вектор-строку в схеме $[XYZ]:[\{a_1\}\{b_1\}\{c_1\}]$.

Теперь, по-прежнему, считаем, что переменная Z принимает значение c_1 и будем рассматривать вторую подзадачу: $R_3[WZ] \oplus \oplus R_4[LW] \oplus R_6[ML] \oplus R_5[ZM]$.

Рассматривая ограничение $R_3[WZ]$, с учетом нового домена переменной Z, заключаем, что само ограничение преобразуется в вектор-строку $[\{d_1\}\{c_1\}]$, а домен переменной W сужается до одноэлементного множества $\{d_1\}$. При активации ограничения $R_4[LW]$, оно преобразуется в вектор-строку $[\{e_1\}\{d_1\}]$, а домен переменной L сужается до $\{e_1\}$. После этого анализируется ограничение $R_6[ML]$, которое упрощается до вектор-строки $[\{f_1, f_6\}\{e_1\}]$. Домен переменной M сужается до $\{f_1, f_6\}$. Наконец, активируется ограничение $R_5[ZM]$, которое с учетом уже конкретизированных доменов представляет собой вектор-строку $[\{c_1\}\{f_1\}]$.

Движение в противоположном направлении дает нам вектор-строку в схеме $[ZWLM]:[\{c_1\}\{d_1\}\{e_1\}\{f_1, f_6\}]$.

Итак, мы осуществили элементарный шаг алгоритма. Все остальные шаги для оставшихся пяти присваиваний выполняются по аналогии. Результаты всех шагов можно сгруппировать в две *C*-системы:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} X \\ \left[\begin{array}{c} \{a_1\} \\ \{a_1\} \\ \{a_2\} \\ \{a_3\} \\ \{a_2\} \\ \{a_2\} \\ \{a_4\} \\ \{a_5\} \\ \{a_6\} \end{array}\right. \end{array} \begin{array}{c} Y \\ \begin{array}{c} \{b_1\} \\ \{b_2\} \\ \{b_3\} \\ \{b_6\} \\ \{b_4\} \\ \{b_5\} \\ \{b_7\} \\ \{b_8\} \\ \{b_9\} \end{array} \end{array} \begin{array}{c} Z \\ \left.\begin{array}{c} \{c_1\} \\ \{c_2\} \\ \{c_2\} \\ \{c_2\} \\ \{c_3\} \\ \{c_4\} \\ \{c_4\} \\ \{c_5\} \\ \{c_6\} \end{array}\right] \end{array},$$

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} Z \\ \left[\begin{array}{c} \{c_1\} \\ \{c_2\} \\ \{c_2\} \\ \{c_4\} \\ \{c_6\} \end{array}\right. \end{array} \begin{array}{c} W \\ \begin{array}{c} \{d_1\} \\ \{d_2\} \\ \{d_2\} \\ \{d_4\} \\ \{d_6\} \end{array} \end{array} \begin{array}{c} L \\ \begin{array}{c} \{e_1\} \\ \{e_2\} \\ \{e_3\} \\ \{e_6\} \\ \{e_7\} \end{array} \end{array} \begin{array}{c} M \\ \left.\begin{array}{c} \{f_1\} \\ \{f_3, f_4\} \\ \{f_4\} \\ \{f_9\} \\ \{f_{13}\} \end{array}\right] \end{array}.$$

Первая из приведенных *C*-систем описывает множество решений для первой подзадачи CSP, вторая — для второй подзадачи CSP.

Теперь можно соединить данные две *C*-системы (по общей переменной Z), получив результирующую *C*-систему, описывающую все решения исходной задачи. Соединение двух *C*-систем выполняется в полиномиальное время с применением Теоремы 1.

Далее рассмотрим второй из разработанных гибридных методов.

3. ГИБРИДНЫЙ МЕТОД, СОЧЕТАЮЩИЙ РАСПРОСТРАНЕНИЕ НЕЧИСЛОВЫХ ОГРАНИЧЕНИЙ И ЛОКАЛЬНЫЙ ПОИСК

Известно, что методы, реализующие распространение ограничений, и методы, реализующие процедуры локального поиска, обладают низкой вычислительной сложностью. Ввиду, этого представляется перспективной интеграция методов данных классов в гибриды. Однако, известно, что методы распространения ограничений используют частичные присваивания (означиваются лишь некоторые переменные), чтобы на основе конкретизированных значений некоторых переменных делать выводы о сужении (уточнении) областей определения оставшихся переменных, а методы локального поиска используют полные присваивания (значения принимают все переменные), которые либо признаются допустимыми, либо особым образом модифицируются для получения допустимых присваиваний.

В статье предложен гибридный метод, интегрирующий следующие компоненты:

1) авторские методы распространения нечисловых ограничений (методы вывода на *D*-системах) для сокращения размерности пространства поиска;

2) метод локального поиска на частичных присваиваниях, основанный на применении эвристики с минимальными конфликтами для быстрого выхода из подпространств, не содержащих решение;

3) поиск с запретами (tabu search) [17] для избегания повторного прохождения уже исследованных состояний.

Правила вывода на D-системах

Приведем без доказательств утверждения, используемые для организации вывода на *D*-системах [5]:

Утверждение 1' (*У1'*). Если хотя бы одна строка *D*-системы пуста (содержит все пустые компоненты), то *D*-система пуста (соответствующая система ограничений несовместна, задача CSP не имеет решения).

Утверждение 2' (*У2'*). Если все компоненты некоторого атрибута пусты, то данный атрибут можно удалить из *D*-системы (удаляются все компоненты, стоящие в соответствующем столбце).

Утверждение 3' (*У3'*). Если в *D*-системе есть строка (кортеж), содержащая лишь одну непустую компоненту, то все значения, не входящие в эту компоненту, удаляются из соответствующего домена.

Утверждение 4' (*У4'*). Если строка *D*-системы содержит хотя бы одну полную компоненту, то она удаляется (можно удалить соответствующее ограничение из системы ограничений).

Утверждение 5' (*У5'*). Если компонента атрибута *D*-системы содержит значение, не принадлежащее соответствующему домену, то это значение удаляется из компоненты.

Утверждения 1'–5' позволяют исключать «лишние» значения из отдельных компонент, из доменов переменных (атрибутов), элиминировать строки и/или столбцы матриц ограничений, «сужая» область поиска и ускоряя получение решений задачи CSP. На основе перечисленных утверждений были модифицированы известные алгоритмы достижения дуговой и вершинной совместностей для случая нечисловых ограничений [7].

Подход к организации локального поиска на основе анализа compressed-таблиц

Методы локального поиска — основные методы решения сложных задач CSP. Идея локального поиска заключается в том, чтобы начиная с произвольно сгенерированного кандидата в решения проблемы (полного присваивания), пошагово его улучшать, например, путем уменьшения количества неу-

довлетворенных ограничений (конфликтов). Алгоритмы локального поиска отличаются методами нахождения этого улучшения. Один из недостатков алгоритмов локального поиска — их неполнота, т. е. в них поиск может остановиться в локальном оптимуме, который в действительности не является глобальным решением.

В статье [8] был предложен подход к организации процедур локального поиска в задачах удовлетворения ограничений, основанный на «сжатом» представлении качественных ограничений в виде *D*-систем.

При интерпретации понятия конфликта при организации процедур локального поиска на *D*-системах, конфликтом считается ситуация, когда в процессе присваивания значений всем переменным *D*-системы в ней образуется пустая строка — строка, состоящая полностью из пустых компонент. В методах систематического поиска обнаружение хотя бы одной пустой строки означает, что ограничение неразрешимо и необходимо выполнять откат/возврат. В рамках предлагаемого подхода к организации локального поиска требуется произвести подсчет количества конфликтов для принятия решения о дальнейшем продвижении к соседним присваиваниям.

Теперь уточним понятие соседнего состояния. Соседнее состояние — это полное присваивание, которое отличается от текущего лишь значением одной переменной.

Процесс локального поиска сводится к переходу от текущего состояния к соседнему до тех пор, пока не будет выполнено ограничение (*D*-система) или не израсходовано число попыток. Среди всех возможных соседних состояний выбирается то, которое наиболее быстро может привести к цели. Для этого рекомендуется применять следующие эвристики. Рекомендуется выбирать из конфликтного множества переменную, участвующую в наибольшем количестве конфликтов. Переменная включается в конфликтное множество переменных, если присваивание в этом атрибуте является одной из причин образования хотя бы одной пустой строки. Рекомендуется выбирать наиболее часто встречающееся значение в соответствующем столбце.

Также в [8] рассмотрен способ ускорения методов локального поиска на compressed-таблицах. Это ускорение основано на применении следствий из теоремы, приведенной ниже.

Данная теорема опирается на отношения частичного порядка, которые определяются для значений внутри домена каждого атрибута. Поэтому сначала напомним, что, если в *D*-системе со схемой S и множеством номеров кортежей $\{n_m\}$ выбрать некоторый атрибут X (X входит в схему S) с доменом D_X, то для значений из D_X отношение частичного порядка «≤» вводится следующим образом: $a \le b$, тогда и только тогда, когда $\{n_0\} \subseteq \{n_p\}$, где: $a, b \in D_X$; а $\{n_0\}$ и $\{n_p\}$ есть множества номеров строк *D*-системы, компоненты которых в атрибуте X содержат значения a и b, соответственно. Проще говоря, одно значение доминирует другое в рамках одного домена, если оно встречается в данном столбце во всех тех строках, что и первое, и еще в каких-то других строках. На рис. 3. приведены диаграммы Хассе для значений атрибутов X и Y некоторой *D*-системы.

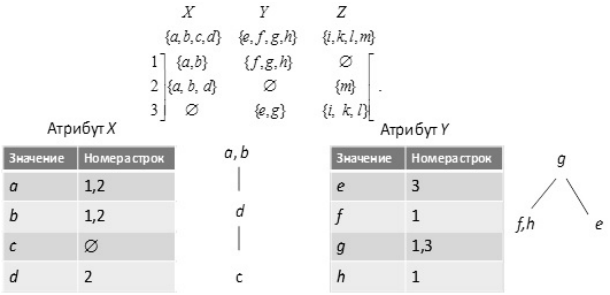

Рис. 3. Отношения частичного порядка на значениях доменов атрибутов исходной D-системы
[Fig. 3. Partial order relations on the attribute domain values of the original D-system]

Так для атрибута Y значение g доминирует значение e (то есть, $g \ge e$), поскольку оно также как и e присутствует в 3 стоке второго столбца, но встречается и без e в первой строке второго столбца. В целом, для атрибута Y имеем: $e, f, h \le g$. Для атрибута X выполняется $d \le a, b$.

Теорема 2. *Если в D-системе со схемой S для некоторого атрибута X (X входит в*

схему S) с доменом D_X имеет место: $a \le b$, то для любой пары полных присваиваний $\{(X_1,c_1),...,(X_p,a),...,(X_n,c_n)\}$ и $\{(X_1,c_1),...,(X_p,b),...,(X_n,c_n)\}$ количество конфликтов, порождаемых вторым присваиванием не больше, чем количества конфликтов, порождаемых первым присваиванием.

Следствие 2.1. Если некоторое значение *b*, атрибута X_p входит в недопустимое присваивание, то при замене этого значения на другое значение *a*, атрибута X_p, такое что $a \le b$, также получится недопустимое присваивание.

Следствие 2.2. Если некоторое значение *a* атрибута X_p входит в допустимое присваивание, то при замене этого значения на другое значение *b* атрибута X_p, такое что $a \le b$, также получится допустимое присваивание.

Далее рассмотрим типовые случаи, когда применение теоремы и следствий из неё способно ускорить поиск.

Во-первых, можно изначально генерировать присваивания, содержащие доминирующие значения в каждом атрибуте. Если так поступить в рассматриваемом нами примере, то будет сгенерировано, например, следующее присваивание: $\{(X,a),(Y,g),(Z,l)\}$.

Оно удовлетворяет *D*-системе, приведенной на рис. 3, поэтому является решением.

Во-вторых, если получено некоторое решение, то на основании Следствия 2.2 можно расширить его до подпространства в пространстве допустимых присваиваний. Так, если присваивание $\{(X,a),(Y,g),(Z,l)\}$ является решением, то, учитывая отношения частичного порядка на доменах переменных получим подпространство допустимых присваиваний, $X \to \{a,b\}$, $Y \to \{g\}$, $Z \to \{i,k,l\}$. В частности, одним из выводимых решений будет: $\{(X,b),(Y,g),(Z,l)\}$.

В-третьих, если в текущем состоянии имеются конфликты, то при переходе к следующему состоянию в указанном конфликтном атрибуте приоритет следует отдавать одному из допустимых на шаге выбора доминирующих значений. Предложенная ранее эвристика, предписывающая выбирать наиболее часто встречающееся в столбце значение, является своеобразной аппроксимацией данного правила.

В-четвертых, если обнаружено недопустимое присваивание, то его можно расширить на основе Следствия 2.1 до подпространства недопустимых присваиваний. Пусть на текущем шаге получено недопустимое присваивание $\{(X,c),(Y,g),(Z,l)\}$, которое порождает конфликт во второй строке. Тогда, учитывая отношения частичного порядка на доменах переменных, имеем следующую область в пространстве недопустимых присваиваний: $X \to \{c\}$, $Y \to \{e,f,h,g\}$, $Z \to \{i,k,l\}$.

Схема гибридного метода, интегрирующего локальный поиск и распространение нечисловых ограничений

Разработанный гибридный метод основан на представлении задачи удовлетворения ограничений в виде *D*-системы и использует эвристику подсчета конфликтов для модификации текущего несовместного частичного присваивания. Вектор присваиваний предложено моделировать в виде вектор-строки *C*-типа, компоненты которого могут содержать не только конкретное значение, но и подмножество допустимых значений данной переменной. На этапе расширения текущего частичного присваивания выбирается переменная с минимальным доменом, затем выбирается то значение переменной, которое наиболее часто встречается в соответствующем столбце. По сути, разработанный метод является модификацией известного метода tabu decision-repair [18, 19] на задачу обработки предлагаемых compressed-таблиц.

Схема работы метода состоит в следующем:

1. Для домена каждой переменной устанавливается частичный порядок на значениях переменных (как в предлагаемом выше алгоритме локального поиска).

2. Частичные присваивания расширяются итеративно до тех пор, пока a) не будет обнаружена несовместность (тогда переход к шагу 3 алгоритма); b) не будет найдено допустимое присваивание (переход к шагу 4) или c) израсходовано количество попыток (переход к шагу 5). При этом применяются следующие эвристики для выбора переменной и значения переменной: выбирается переменная с

минимальным доменом, затем выбирается то значение переменной, которое наиболее часто встречается в соответствующем столбце. В ходе выполнения присваиваний редуцируется и сама *D*-система согласно ранее описанным правилам редукции.

3. При обнаружении несовместного частичного (или полного присваивания), вместо осуществления процедуры backtracking, производится ранее предложенная процедура локального поиска. Несовместное частичное присваивание на основании Следствия 2.1 достраивается до области в пространстве недопустимых частичных присваиваний для заданного набора переменных. Обнаруженные конфликты (присваивания, приведшие к образованию пустых строк *D*-системы) регистрируются в специальной памяти, организованной по принципу очереди. Целесообразно ограничить очередь небольшим числом элементов.

На основе подсчета количества конфликтов (пустых строк *D*-системы) выбирается переменная, участвующая в максимальном количестве конфликтов. Модификация частичного присваивания (переход в соседнее состояние) осуществляется путем вычисления дополнения той компоненты частичного присваивания, которая соответствует выбранной конфликтной переменной. Переход к шагу 2.

4. Полученное допустимое присваивание расширяется на основе Следствия 2.2 до области в пространстве допустимых присваиваний.

5. Конец.

ЗАКЛЮЧЕНИЕ

Впервые разработаны гибридные методы решения задач удовлетворения нечисловых ограничений, формализованных с помощью *C*- и *D*-систем. Отличительной чертой предлагаемых методов является то, что они не используют стратегий поиска в глубину с возвратами, приводящих, зачастую, к экспоненциальному росту времени выполнения вычислительных процедур при росте размерности задачи.

Первый из предложенных гибридных методов интегрирует авторские методы распространения нечисловых ограничений с существующими алгоритмами структурной декомпозиции графа ограничений. Основная вычислительная сложность данного метода заключена в компоненте, реализующей структурную декомпозицию графа ограничений, но имеются приближенные («жадные») алгоритмы, позволяющие достаточно быстро осуществлять подобную декомпозицию. Каждая подзадача, имеющая древовидную структуру, решается за полиномиальное время с помощью авторских алгоритмов распространения нечисловых ограничений. Метод хорошо подходит для ситуации, когда имеется серия задач с одной и той же структурой графа ограничений. Такая ситуация часто возникает при анализе типовых запросов к хранилищам данных.

Второй из разработанных гибридных методов интегрирует следующие компоненты: авторские методы распространения нечисловых ограничений для сокращения размерности пространства поиска; метод локального поиска на частичных присваиваниях для быстрого выхода из подпространств, не содержащих решение; поиск с запретами (tabu search) для избегания повторного прохождения уже исследованных состояний. Он предназначен для ситуаций, когда требуется найти решение на основе больших объемом информации, но при этом отсутствуют априорные сведения о структуре графа задачи CSP. Метод хорошо подходит для задач планирования и составления расписаний.

Применение специализированных структур для представления и обработки табличных ограничений позволяет сократить расходы памяти на представление задачи, а детальный анализ свойств *C*- и *D*-систем лежит в основе предлагаемых способов ускорения вычислительных процедур.

Исследование выполнено при финансовой поддержке РФФИ в рамках научных проектов № 18-07-00615а, 20-07-00708а.

КОНФЛИКТ ИНТЕРЕСОВ

Автор декларирует отсутствие явных и потенциальных конфликтов интересов, связанных с публикацией настоящей статьи.

СПИСОК ЛИТЕРАТУРЫ

1. *Katsirelos, G.* A Compression Algorithm for Large Arity Extensional Constraints / G. Katsirelos, T. Walsh // Principles and Practice of Constraint Programming. Lecture Notes in Computer Science, Bessière C. (eds). – Berlin, Heidelberg: Springer. – 2007. – vol 4741. pp 379–393. doi:10.1007/978-3-540-74970-7_28.

2. *Cheng, K. C.* An MDD-based generalized arc consistency algorithm for positive and negative table constraints and some global constraints / K. C. Cheng, R. H. Yap // Constraints. – 2010. – vol. 15(2). pp. 265–304. doi: 10.1007/s10601-009-9087-y.

3. *Wang, R.* Optimizing Simple Tabular Reduction with a bitwise representation. / R. Wang, [etc.] // Proceedings of IJCAI. – 2016. – P. 787–793.

4. *Verhaeghe, H.* Extending Compact-Table to Basic Smart Tables / H. Verhaeghe, [etc.] // Proceedings of 23rd International Conference, Principles and Practice of Constraint Programming., CP 2017, Melbourne, VIC, Australia, August 28 – September 1. – 2017. – P. 297-307. doi:10.1007/978-3-319-66158-2_19.

5. *Зуенко, А. А.* Вывод на ограничениях с применением матричного представления конечных предикатов / А. А. Зуенко // Искусственный интеллект и принятие решений. – 2014. – № 3. – С. 21–31.

6. *Zuenko, A. A.* Application of constraint propagation techniques to speed up processing of queries to ontologies. / A. A. Zuenko, P. A. Lomov, A. G. Oleinik // SPIIRAS Proceedings. – 2017. – 1(50). – P. 112 136. doi:10.15622/sp.50.5.

7. *Zuenko, A.* Matrix-like Structures for Representation and Processing of Constraints Over Finite Domains / A. Zuenko // Proceedings of the Third International Scientific Conference “Intelligent Information Technologies for Industry” (IITI’18, Volume 2) In: Advances in Intelligent Systems and Computing (AISC). Abraham et al. (Eds.). Springer Nature Switzerland AG. – 2019. – Vol. 875. – P. 428-438. doi:10.1007/978-3-030-01821-4_45.

8. *Zuenko, Alexander A.* Local Search in Solution of Constraint Satisfaction Problems Represented by Non-Numerical Matrices / Alexander A. Zuenko // Proceedings of the 2nd International Conference on Computer Science and Application Engineering (CSAE ‘18), October 22-24, 2018, Hohhot, China. ACM New York, NY, USA ©2018. – 2018. Article 138. doi:10.1145/3207677.3277959.

9. *Кулик, Б. А.* Алгебраический подход к интеллектуальной обработке данных и знаний / Б. А. Кулик, А. А. Зуенко, А. Я. Фридман. – СПб. : Изд-во Политехн. ун-та, 2010. 235 с.

10. *Russel, S.* Artificial Intelligence: A Modern Approach. 3rd edition / S. Russel, P. Norvig – Prentice Hall. 2010.

11. Miguel, I. Solution techniques for constraint satisfaction problems: foundations / I. Miguel, Q. Shen // Artificial Intelligence Review. – 2001. – № 15. – P. 241–265. doi:10.1023/a:1011096320004.

12. *Dechter, R.* Tree clustering for constraint networks / R. Dechter, J. Pearl // Artificial Intelligence. – 1989. – 38 (3). – P. 353–366. doi:10.1016/0004-3702(89)90037-4.

13. *Dechter, R.* Enhancement schemes for constraint processing: Backjumping, learning and cutset decomposition / R. Dechter // Artificial Intelligence. – 1990. – 41. – P. 273–312. doi: 10.1016/0004-3702(90)90046-3.

14. *Dechter, R.* Network-based heuristics for constraint satisfaction problems / R. Dechter, J. Pearl // Artificial Intelligence. – 1987. – 34 (1). – P. 1–38. doi:10.1016/0004-3702(87)90002-6.

15. *Freuder, E. C.* A sufficient condition for backtrack-free search / E. C. Freuder // Journal of the ACM. – 1982. – 29 (1). – P. 24–32. doi:10.1145/322290.322292.

16. *Freuder, E. C.* A sufficient condition for backtrack-bounded search / E. C. Freuder // Journal of the ACM. – 1985. – 32 (4). – P. 755–761. doi:10.1145/4221.4225.

17. *Steinmann, O.* Tabu search vs. random walk / O.Steinmann, A.Strohmaier, T.St¨utzle // KI-97: Advances in Artificial Intelligence,

Springer Verlag, Berlin, Germany. – 1997. – P. 337–348. doi:10.1007/3540634932_27.

18. *Jussien, N.* Local search with constraint propagation and conflict-based heuristics. / N. Jussien, O. Lhomme // Artificial Intelligence. – 2002. – vol. 139. – P. 21-45. doi:10.1016/s0004-3702(02)00221-7.

19. *Chatzikokolakis, K.* Construction and Repair: A Hybrid Approach to Search in CSPs. / K. Chatzikokolakis, G. Boukeas, P. Stamatopoulos / G. A. Vouros, T. Panayiotopoulos (Eds.). – SETN. 2004, – LNAI 3025, 2004. – P. 342–351. doi: 10.1007/978-3-540-24674-9_36.

Зуенко Александр Анатольевич — канд. техн. наук, ведущий научный сотрудник, Институт информатики и математического моделирования – обособленное подразделение Федерального государственного бюджетного учреждения Федерального исследовательского центра «Кольский научный центр Российской академии наук» (ИИММ КНЦ РАН).
E-mail: zuenko@iimm.ru
ORCID iD: https://orcid.org/0000-0002-7165-6651

DOI: https://doi.org/10.17308/sait.2020.3/3041 ISSN 1995-5499
Received 02.06.2020
Accepted 30.09.2020

HYBRID METHODS OF SOLVING NON-NUMERICAL CONSTRAINT SATISFACTION PROBLEMS USING SPECIALIZED TABLES

© 2020 A. A. Zuenko✉

Institute of Informatics and Mathematical Modeling
«Kola Science Centre of the Russian Academy of Sciences»
24A, Fersman Street, 184209 Apatity, Murmansk Region, Russian Federation

Annotation. To increase the efficiency of representation and processing of qualitative domain dependencies within the framework of constraint programming technology, the article proposes to use specialized table structures: C- and D-systems. From the point of view of constraint programming technologies, these structures can be assigned to such a class of global constraints as compressed tables. For the first time, hybrid methods have been developed for solving problems of satisfaction of non-numerical constraints formalised with the help of the proposed C- and D-systems. A distinctive feature of the author's methods is that they do not use depth-first search strategies, which lead to an exponential increase in the execution time of the computational procedures with an increase in the dimension of the problem. The first of the proposed hybrid methods integrates the author's methods for non-numerical propagation of constraints with existing algorithms for the structural decomposition of the constraint graph. As a result of decomposition, the problem is divided into sub-problems, the constraint graph of each of which is a tree. In this case, the sub-problems are solved in polynomial time using the author's algorithms for the propagation of non-numeric constraints. The method is well suited for situations where there is a series of problems with the same structure of the constraint graph, for example, when analysing typical queries for data warehouses. The second of the developed hybrid methods integrates the following components: original methods of non-numeric constraints propagation to reduce the dimension of the search space; a method of local search on partial assignments to quickly exit non-solution subspaces; a tabu search to avoid repeated passage of already investigated states. It is intended for situations when you need to find a solution based on a large amount of information, but there is no a priori information about the graph structure of the constraint satisfaction problem.
Keywords: constraint satisfaction problem, local search, constraint propagation, structural decomposition.

✉ Zuenko Alexander A.
e-mail: zuenko@iimm.ru

CONFLICT OF INTEREST

The author declare the absence of obvious and potential conflicts of interest related to the publication of this article.

REFERENCES

1. *Katsirelos G., Walsh T.* A Compression algorithm for large arity extensional constraints. In: Bessière C. (eds) Proceedings of Principles and Practice of Constraint Programming – CP 2007. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer. 2007. 4741. 379-393. DOI: 10.1007/978-3-540-74970-7_28.

2. *Cheng K. C., Yap R. H.* An MDD-based generalized arc consistency algorithm for positive and negative table constraints and some global constraints. Constraints, 2010. 15(2). 265–304. DOI: 10.1007/s10601-009-9087-y

3. *Wang R., Xia W., Yap R. H. C., Li Z.* Optimizing simple tabular reduction with a bitwise representation. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI), 2016. 787–793.

4. *Verhaeghe H., Lecoutre C., Deville Y., Schaus P.* Extending compact-table to basic smart tables. In: Proceedings of 23rd International Conference Principles and Practice of Constraint Programming (CP 2017). Melbourne, VIC, Australia, August 28 – September 1. 2017. 297–307. DOI: 10.1007/978-3-319-66158-2_19.

5. *Zuenko A. A.* Inference on constraints using the matrix representation of finite predicates. Artificial Intelligent and Decision Making, 2014. 3. 21–31. (in Russian).

6. *Zuenko A. A.* Application of constraint propagation techniques to speed up processing of queries to ontologies. SPIIRAS Proceedings, 2017. 1(50). 112 136. DOI: 10.15622/sp.50.5.

7. *Zuenko A.* Matrix-like structures for representation and processing of constraints over finite domains. In: Abraham et al. (Eds.). Advances in Intelligent Systems and Computing (AISC). Springer Nature Switzerland AG, 2019. 875. 428-438. DOI: 10.1007/978-3-030-01821-4_45.

8. *Zuenko A. A.* Local search in solution of constraint satisfaction problems represented by non-numerical matrices. In: Proceedings of the 2nd International Conference on Computer Science and Application Engineering (CSAE '18), October 22-24, 2018, Hohhot, China. ACM New York, NY, USA. 2018. Article 138. DOI: 10.1145/3207677.3277959.

9. *Kulik B., Zuenko A., Fridman A.* Algebraic approach to the intelligent processing of data and knowledge. Saint Petersburg, Izd-vo Politekhn. un-ta. 2010. 235 p. (in Russian).

10. *Russel S., Norvig P.* Artificial intelligence: a modern approach. 3rd edition. Prentice Hall. 2010. 1132 p.

11. *Miguel I., Shen Q.* Solution techniques for constraint satisfaction problems: foundations. Artificial Intelligence Review, 2001. 15. 241–265. DOI: 10.1023/a:1011096320004.

12. *Dechter R., Pearl J.* Tree clustering for constraint networks. Artificial Intelligence, 1989. 38 (3). 353–366. DOI: 10.1016/0004-3702(89)90037-4.

13. *Dechter R.* Enhancement schemes for constraint processing: backjumping, learning and cutset decomposition. Artificial Intelligence, 1990. 41. 273–312. DOI: 10.1016/0004-3702(90)90046-3.

14. *Dechter R.* Network-based heuristics for constraint satisfaction problems. Artificial Intelligence, 1987. 34 (1). 1–38. DOI: 10.1016/0004-3702(87)90002-6.

15. *Freuder E. C.* A sufficient condition for backtrack-free search. Journal of the ACM, 1982. 29 (1). 24–32. DOI: 10.1145/322290.322292.

16. *Freuder E. C.* A sufficient condition for backtrack-bounded search. Journal of the ACM, 1985. 32 (4). 755–761. DOI: 10.1145/4221.4225.

17. *Steinmann O., Strohmaier A.,* St¨utzle T. Tabu search vs. random walk. In: Brewka G., Habel C., Nebel B. (eds) Proceedings of KI-97: Advances in Artificial Intelligence. Springer Verlag, Berlin, Germany. 1997. 337-348. DOI: 10.1007/3540634932_27.

18. *Jussien N., Lhomme O.* Local search with constraint propagation and conflict-based heuristics. Artificial Intelligence, 2002. 139. 21-45. DOI: 10.1016/s0004-3702(02)00221-7.

19. *Chatzikokolakis K., Boukeas G., Stamatopoulos P.* Construction and repair: a hybrid approach to search in CSPs. In: G. A. Vouros, T. Panayiotopoulos (Eds.). Methods and Ap-

plications of Artificial Intelligence. SETN 2004. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2004. 3025. 342–351. DOI: 10.1007/978-3-540-24674-9_36.

Zuenko Alexander A. — PhD in Technical Sciences, leading researcher, Institute of Informatics and Mathematical Modelling, Subdivision of the Federal Research Centre "Kola Science Centre of the Russian Academy of Sciences".
E-mail: zuenko@iimm.ru
ORCID iD: https://orcid.org/0000-0002-7165-6651