А.А. Зуенко

Вывод на ограничениях с применением матричного представления конечных предикатов

Аннотация. В работе предлагаются методы распространения ограничений и эвристического поиска, направленные на решение задач удовлетворения ограничений, где переменные определены на конечных доменах. Разработанные методы используют матричное представление конечных предикатов, что позволяет обойтись без преобразования ограничений в совокупность бинарных отношений и дает возможность эффективно редуцировать пространство поиска.

Ключевые слова: задача удовлетворения ограничений, алгебра кортежей, редукция пространства поиска, эвристический поиск, комбинаторный поиск.

Введение

В искусственном интеллекте технология программирования в ограничениях широко используется при решении многих прикладных задач, таких как планирование и календарное планирование, задачи назначения частоты, обработка изображений, тестирование сверхбольших интегральных схем, анализ языков программирования и понимания естественного языка и т.д. [1].

Решение многих практических задач сопряжено с анализом как количественных, так и качественных ограничений. В этом случае стандартные численные методы или чисто логические методы оказываются неприменимыми. Возникает необходимость в разработке новых интегрированных методов, предполагающих обработку разнородных ограничений. Такую интеграцию удобно выполнять в рамках технологии программирования в ограничениях.

Согласно [2] *задача удовлетворения ограничений* (Constraint Satisfaction Problem – CSP) определена множеством *переменных* x_1, x_2, ..., x_n, и множеством *ограничений* C_1, C_2, ..., C_m. Каждая переменная x_i имеет непустую *область определения* D_i (*область возможных значений*, *домен*). Каждое ограничение C_i включает некоторое подмножество переменных и задает допустимые комбинации значений для этого подмножества. Состояние задачи описывается как *присваивание* значений некоторым или всем этим переменным $\{x_i=v_i, x_j=v_j, ...\}$. Присваивание, которое не нарушает никаких ограничений, называется *совместимым*, или *допустимым* присваиванием. *Полным* называется такое присваивание, в котором участвует каждая переменная. *Решением задачи* CSP является полное присваивание, которое удовлетворяет всем ограничениям.

Любая задача CSP может быть сформулирована как стандартная задача поиска. В частности, ей может быть дана инкрементная формулировка, если положить, что:

- начальное состояние – пустое присваивание {}, где ни одной переменной не присвоено значение;
- функция определения преемника позволяет присваивать значение любой переменной с неприсвоенным значением, при условии, что присваиваемое значение данной переменной не будет конфликтовать с присвоенными ранее значениями других переменных;

[1] Работа выполнена при финансовой поддержке РФФИ (проекты №№ 12-07-00550-а, 12-07-00689-а, 13-07-00318-а, 14-07-00205-а, 14-07-00256-а, 14-07-00257-а), ОНИТ РАН (проект 2.3 в рамках текущей Программы фундаментальных научных исследований) и Президиума РАН (проект 4.3 Программы № 16).

- проверка цели состоит в установлении того, является ли текущее присваивание полным;
- стоимость пути постоянна (например, 1) для каждого этапа.

Каждое решение должно представлять собой полное присваивание и поэтому, в худшем случае, должно находиться на глубине n, если имеется n переменных. Другими словами, дерево поиска имеет конечную глубину n (по количеству переменных). По этим причинам для решения задач CSP широко применяются алгоритмы поиска в глубину. В таких задачах сам путь, по которому достигается некоторое решение, не представляет интереса.

Далее рассматриваются только задачи CSP, где переменные имеют конечные области определения. По оценке Bartak [3], вероятно, более 95% приложений задач удовлетворения ограничений используют конечные домены. К такому виду относятся: задача раскраски карты, задача расстановки ферзей на шахматной доске, криптоарифметические задачи, задача составления кроссвордов, расписаний и т.д. Если максимальный размер области определения любой переменной в задаче CSP равен d, то количество возможных полных присваиваний измеряется величиной $O(d^n)$, т.е. зависит экспоненциально от количества переменных, что заставляет применять методы ускорения комбинаторного поиска. К категории задач CSP с конечной областью определения относятся булевы задачи CSP, где переменные могут иметь значения либо *true*, либо *false*. Булевы задачи CSP включают в качестве частных случаев некоторые NP-полные задачи, такие как 3SAT.

Методы решения задач удовлетворения ограничений могут быть разбиты на три класса [4]. Первый класс содержит различные варианты алгоритмов *поиска в глубину с возвратами*, которые строят решение путем расширения частичного присваивания шаг за шагом, используя различные эвристики и применяя разумные стратегии возврата из тупиковых вершин. Ко второму классу относятся алгоритмы *распространения ограничений*, которые исключают из пространства поиска некоторые элементы, не входящие в решение, обеспечивая снижение размерности задачи. Эти алгоритмы не строят сами по себе решение, поскольку исключают не все элементы, не входящие в решение. Они применяются или для препроцессинга задачи до использования алгоритмов другого типа, или перемежаются с шагами алгоритма другого типа (например, поиска с возвратами) для повышения производительности последнего. Наконец, *структурные алгоритмы* используют информацию о структуре первичного или двойственного графа ограничений задачи. Алгоритмы этого класса производят декомпозицию исходной задачи CSP на слабо связанные подзадачи, которые могут быть решены с помощью методов предыдущих двух классов. Таким образом, структурные алгоритмы могут также использоваться в сочетании с алгоритмами других типов.

Задачи CSP, как правило, решаются с использованием *графов* или *гиперграфов ограничений.* Традиционно при решении задач CSP ограничения с конечной областью определения сводятся к множеству бинарных отношений посредством введения достаточного количества вспомогательных переменных.

В настоящей работе для эффективного решения задач в рамках концепции программирования в ограничениях предлагается применять подход, основанный на использовании матричного представления логических ограничений. Под логическими будем понимать ограничения, представляющие собой логические выражения над пропозициональными переменными и/или конечными предикатами. Предлагаемые далее методы принадлежат к первым двум классам методов комбинаторного поиска и позволяют обойтись без предварительной “бинаризации” ограничений.

Прежде чем приступить к непосредственному изложению заявленных методов, приведем краткое описание структур, применяемых для матричного представления логических ограничений. В качестве математического аппарата для моделирования ограничений используется алгебра кортежей (АК) [5, 6].

1. Матрицеподобные структуры для представления конечных предикатов

В [5, 7-9] приводятся основы АК и демонстрируется ее применение для унификации представления и обработки различных видов данных и знаний, а также решения задач логического и логико-вероятностного анализа.

Близкий подход применяется также в [10] для решения задач распознавания образов и упрощения баз знаний. В силу наличия большого количества публикаций и докладов, освещающих особенности алгебраического подхода, используемого в АК, в данной работе в нестрогом изложении приводятся лишь базовые понятия, которые понадобятся в дальнейшем.

Конечные предикаты в АК можно сжато представить с помощью двух типов структур: *C*-систем и *D*-систем. Конкретные экземпляры этих структур называются АК-объектами.

C-система записывается в виде матрицы, состоящей из компонент-множеств и заключенной в прямые скобки, например,

$$T[XYZ]=\begin{bmatrix} A & * & C \\ D & E & * \end{bmatrix}$$ есть *C*-система.

Слева от знака равенство находится заголовок *C*-системы, содержащий имя *C*-системы и *схему*, в которой перечислены имена ее атрибутов (в данном случае *X*, *Y*, *Z*). Подобно тому, как атрибут отношения в реляционной алгебре соотносится с определенным столбцом, в АК атрибут соответствует столбцу *C*-системы. Будем считать, что доменами атрибутов *X*, *Y*, *Z* являются множества D_1, D_2, D_3, соответственно. Справа от знака равенства находится тело *C*-системы. В отличие от реляционных таблиц, каждый кортеж *C*-системы состоит не из отдельных значений, а из компонент-множеств, являющихся подмножествами соответствующего домена.

Например:

$A \subseteq D_1$; $D \subseteq D_1$; $E \subseteq D_2$; $C \subseteq D_3$,

или $A \in 2^{D1}$; $D \in 2^{D1}$; $E \in 2^{D2}$; $C \in 2^{D3}$.

Видно, что компоненты *C*-системы, стоящие в одном и том же столбце, например компоненты *A* и *D*, принадлежат булеану некоторого множества (в нашем случае булеану множества D_1), называемого доменом соответствующего атрибута.

При описании *C*- и *D*-систем используется два вида фиктивных компонент: полная компонента (обозначается «*») – это множество, равное домену соответствующего (по месту ее расположения в кортеже) атрибута; пустая компонента (обозначается «∅») – компонента, не содержащая ни одного значения.

Так запись

$$T[XYZ]=\begin{bmatrix} A & * & C \\ D & E & * \end{bmatrix}$$

для нашего примера эквивалента записи

$$T[XYZ]=\begin{bmatrix} A & D_2 & C \\ D & E & D_3 \end{bmatrix}.$$

Эту *C*-систему можно преобразовать в обычное многоместное отношение следующим образом: $(A \times D_2 \times C) \cup (D \times E \times D_3)$.

С помощью *C*-систем удобно представлять в “сжатом” виде дизъюнктивные нормальные формы (ДНФ) конечных предикатов. Покажем это на примере. Пусть задан конечный предикат:

$\varphi(x, y, z) = (x=a,b) \wedge (y=a,c) \vee (z=d)$.

Для простоты все переменные определены на одном и том же множестве $\{a, b, c, d\}$. Здесь и далее будем использовать запись вида $(x=a,b)$ для обозначения выражения $(x=a) \vee (x=b)$. Учитывая, что область истинности одноместного предиката $(x=a,b)$ есть $\{a,b\}$, то область истинности предиката $\varphi(x, y, z)$ может быть представлена в виде следующей *C*-системы:

$$R[XYZ] = \begin{bmatrix} \{a,b\} & \{a,c\} & * \\ * & * & \{d\} \end{bmatrix}.$$

С помощью *D*-систем моделируются конъюнктивные нормальные формы (КНФ) конечных предикатов. *D*-система записывается как матрица компонент-множеств, которая ограничена перевернутыми прямыми скобками. *D*-системы позволяют легко вычислять дополнение *C*-систем: вычисляется дополнение для каждой компоненты-множества. Например, дополнением *C*-системы *T*[*XYZ*] является *D*-система

$$\overline{T}\,[XYZ]=\rceil\begin{matrix} \overline{A} & \varnothing & \overline{C} \\ \overline{D} & \overline{E} & \varnothing \end{matrix}\lceil .$$

D-систему $\overline{T}$ можно представить в виде обычного отношения по формуле: $((\overline{A} \times D_2 \times D_3) \cup (D_1 \times D_2 \times \overline{C})) \cap ((\overline{D} \times D_2 \times D_3) \cup (D_1 \times \overline{E} \times D_3))$.

Предикат $\neg\varphi = (\neg(x=a,b) \vee \neg(y=a,c)) \wedge \neg(z=d)$, что равносильно $\neg\varphi = ((x=c,d) \vee (y=b,d)) \wedge (z=a,b,c)$, может быть выражен как *D*-система $\overline{R}\,[XYZ]$:

$$\overline{R}\,[XYZ] = \left]\begin{matrix}\{c,d\} & \{b,d\} & \varnothing \\ \varnothing & \varnothing & \{a,b,c\}\end{matrix}\right[.$$

Правила выполнения операций обобщенного объединения ($\cup_G$) и пересечения ($\cap_G$) для *C*- и *D*-структур имеют свою специфику [5] и здесь подробно не разбираются. В частности, пересечение двух *C*-систем сводится к пересечению каждой строки первой *C*-системы с каждой строкой второй. *D*-системы могут быть преобразованы в *C*-системы, например:

$$\overline{T}\,[XYZ] = \left]\begin{matrix}\overline{A} & \varnothing & \overline{C} \\ \overline{D} & \overline{E} & \varnothing\end{matrix}\right[=$$

$$= \begin{bmatrix}\overline{A} & * & * \\ * & * & \overline{C}\end{bmatrix} \cap_G \begin{bmatrix}\overline{D} & * & * \\ * & \overline{E} & *\end{bmatrix} = \begin{bmatrix}\overline{A}\cap\overline{D} & * & * \\ \overline{A} & \overline{E} & * \\ \overline{D} & * & \overline{C} \\ * & \overline{E} & \overline{C}\end{bmatrix}.$$

Результирующая *C*-система (фактически ДНФ) сжато описывает множество всех выполняющих подстановок КНФ, представленной исходной *D*-системой. Однако подобное преобразование слишком трудоемко.

В простейшем случае, если все ограничения удается записать с помощью рассмотренных матрицеподобных структур, в частности *D*-систем, то задачу CSP можно представить в виде задачи о пустоте/непустоте *D*-системы, которая соотносится с задачей выполнимости КНФ конечного предиката, когда ищутся не все выполняющие подстановки, а хотя бы одна из них.

Зачастую поиск решения задачи CSP не ограничивается чисто логическим выводом. Для эффективной организации поиска приходится привлекать семантику предметной области. На практике нередко оказывается, что переменные связаны между собой определенными соотношениями либо носящими нелогический характер, либо порождающими громоздкие конструкции при их формализации с помощью конечных предикатов. Причем, совместное применение логических и численных методов не сводится к их последовательному выполнению. Более того, последовательное применение может приводить к ошибочным результатам, а наиболее эффективные методы на каждом шаге поиска используют как количественные, так и качественные (логические) ограничения, порождая новые классы методов. Далее разбираются разработанные методы исследования пространства поиска, которые основаны на матричном представлении конечных предикатов.

2. Базовые методы вывода на ограничениях

Среди методов поиска в глубину, используемых при решении CSP, можно выделить те методы, где переменные просматриваются в некотором заранее выбранном порядке. Однако такое статическое упорядочение переменных редко приводит к эффективному поиску. Следовательно, выбор переменной и присваиваемого ей значения должен осуществляется динамически на каждом шаге поиска. Для осуществления подобного выбора применяются различного рода *эвристики*. Методы, использующие эвристики, называются методами *информированного* или *эвристического поиска*. Эти методы позволяют отложить исследование менее перспективных ветвей дерева поиска, отдав предпочтение более перспективным с точки зрения скорейшего получения решения.

Помимо применения эвристик, позволяющих более эффективно организовать “разбор случаев”, можно существенно ускорить процесс поиска при решении задач CSP, рассматривая внутреннюю структуру текущего состояния задачи. Зачастую, пространство поиска может быть значительно редуцировано вообще без организации “ветвления” с помощью алгоритмов, позволяющих преобразовать описание текущего состояния в эквивалентное ему более простое описание. Обычно такие алгоритмы имеют полиномиальную оценку временной сложности.

При решении задач CSP дерево поиска можно представить как *дерево частичных присваиваний* – это дерево, где каждая вершина соответствует некоторому частичному присваиванию. Корень дерева отвечает пустому присваиванию. В каждой вершине *v* выбирается лишь одна переменная, которой еще не было присвоено значение на предыдущих уровнях дерева поиска.

Поскольку переменным задачи CSP в АК соответствуют атрибуты, заданные на определенных доменах, то вполне естественно рассматривать процесс присваивания значения переменной как переход от старого домена

атрибута к новому одноэлементному домену. Поясним эту мысль более детально.

Рассмотрим, как реализуется операция присваивания переменной некоторого значения с использованием матричного представления логических ограничений. Пусть у нас изначально задано ограничение в виде *D*-системы

$$C[XYZ] = \begin{bmatrix} \{c\} & \{d,e\} & \{h\} \\ \{a\} & \{e\} & \{g\} \end{bmatrix}.$$

Причем соответствие между атрибутами и их доменами следующее: $X - \{a, b, c\}$, $Y - \{d, e, f\}$, $Z - \{g, h\}$. Текущее присваивание (например, $x=b$) задается *C*-системой, состоящей из одной строки:

$$E[XYZ] = \begin{bmatrix} \{b\} & * & * \end{bmatrix}.$$

Тогда процесс присваивания значения переменной и вычисления "остатка" *D*-системы $C[XYZ]$ выглядит так:

$$C[XYZ] \cap_G E[XYZ] =$$

$$= \begin{bmatrix} \{c\} & \{d,e\} & \{h\} \\ \{a\} & \{e\} & \{g\} \end{bmatrix} \cap_G \begin{bmatrix} \{b\} & * & * \end{bmatrix} =$$

$$= \begin{bmatrix} \{c\} & * & * \\ * & \{d,e\} & * \\ * & * & \{h\} \end{bmatrix} \cap_G \begin{bmatrix} \{a\} & * & * \\ * & \{e\} & * \\ * & * & \{g\} \end{bmatrix} \cap_G$$

$$\begin{bmatrix} \{b\} & * & * \end{bmatrix} = \begin{bmatrix} \{b\} & \{d,e\} & * \\ \{b\} & * & \{h\} \end{bmatrix} \cap_G \begin{bmatrix} \{b\} & \{e\} & * \\ \{b\} & * & \{g\} \end{bmatrix}.$$

Последнее выражение получено в результате применения операции обобщенного пересечения сначала к множителям 1 и 3, затем к множителям 2 и 3.

Пересечение двух полученных *C*-систем уже не удается представить без трудоемких преобразований в виде единой *D*-системы в пространстве $X - \{a, b, c\}$, $Y - \{d, e, f\}$, $Z - \{g, h\}$. Однако если перейти к пространству $X - \{b\}$, $Y - \{d, e, f\}$, $Z - \{g, h\}$ (а такой переход вполне обоснован с учетом того, что результат пересечения в компонентах атрибута X не может иметь значений отличных от b), тогда:

X	Y	Z
$\{b\}$	$\{d,e,f\}$	$\{g,h\}$
$\varnothing$	$\{d,e\}$	$\{h\}$
$\varnothing$	$\{e\}$	$\{g\}$

Для удобства объяснения здесь и далее в верхних двух строках *D*-систем записываются имена атрибутов и множества допустимых значений этих атрибутов (домены атрибутов).

Если сравнить между собой исходную *D*-систему и ту *D*-систему, которая получена после присваивания, то можно сделать важный вывод о том, что операция присваивания сводится к замене старого домена атрибута X на новый одноэлементный домен с вычеркиванием из соответствующего столбца тех значений, которые не принадлежат новому домену. Другими словами, можно обойтись без промежуточных вычислений, производимых с помощью *C*-систем. Требуется лишь исключить некоторые значения из соответствующего домена.

Заметим, что последнюю *D*-систему можно упростить путем элиминации атрибута X, поскольку он более не связан ограничениями с другими атрибутами, предварительно запомнив, что домен атрибута $X - \{b\}$:

Y	Z
$\{d,e,f\}$	$\{g,h\}$
$\{d,e\}$	$\{h\}$
$\{e\}$	$\{g\}$

Теперь настало время уточнить, какое представление состояния задачи CSP используется в разработанных автором методах. Будем считать, что ограничения задачи моделируются в виде некоторой *D*-системы. Состояние задачи в терминах введенных матрицеподобных структур характеризуется с помощью совокупности доменов атрибутов, которые уже элиминированы из *D*-системы, а также самой *D*-системы, представляющей собой остаток, полученный из исходной *D*-системы в ходе присваиваний и применения правил редукции, которые представлены ниже.

Одним из основных отличий методов, предлагаемых в данной работе, от ранее разработанных в АК методов, является то, что в новых методах решение строится путем пошагового усечения доменов. Само решение представляется в виде совокупности доменов или установлении того факта, что все вершины дерева поиска являются тупиковыми.

Далее перейдем к описанию предлагаемых методов поиска решения задачи CSP, опишем используемые эвристики и применяемые правила редукции пространства поиска.

2.1. Выбор наилучшего преемника на основе эвристик

Для выбора “наилучшего” преемника текущей вершины дерева поиска необходимо ответить на вопросы:

1. Какой переменной на данном шаге присваивать значения?
2. Выбор, какого из значений переменной быстрее всего может привести к цели?

Для ответа на данные вопросы в настоящей работе предлагаются эвристики:

Э1. Выбирать атрибут с доменом, содержащим наименьшее количество значений, что позволяет проверять меньшее количество преемников.

Э2. В случае неоднозначности выбора, производимого согласно **Э1**, выбирать атрибут, количество непустых компонент которого максимально.

Э3. Для формирования нового одноэлементного домена выбирать наиболее часто встречающееся в кортежах *D*-системы значение атрибута.

2.2. Редукция пространства поиска

Уточним некоторые особенности поиска на основе матричного представления конечных предикатов, а именно:

1. Каков признак того, что текущая ветка поиска является тупиковой (соответствующее этой вершине текущее присваивание недопустимо)?
2. Как уменьшить размерность пространства поиска, не прибегая к ветвлению?
3. Что служит признаком успешного завершения процесса поиска?

Для ответа на эти вопросы рассмотрим следующие утверждения, приводимые здесь без доказательств:

Утверждение 1 (**У1**). Если строка *D*-системы содержит все пустые компоненты, то *D*-система пуста.

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 (компонента становится новым доменом данного атрибута).

Утверждение 4 (**У4**). Если строка *D*-системы содержит хотя бы одну полную компоненту, то она удаляется.

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

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

Заметим, что утверждение **У5** уже использовалось нами при рассмотрении интерпретации в АК операции присваивания значения очередной переменной.

Ответ на первый из поставленных вопросов дает нам **У1**, т.е. *признаком недопустимости* частичного присваивания является пустота *D*-системы.

Ответом на второй вопрос служат остальные утверждения, часть из которых позволяет исключать значения из доменов атрибутов (**У3, У5**) или даже сами атрибуты (**У2**), а часть исключать из рассмотрения лишние строки (**У4**, **У6**).

Признак успешного завершения процесса поиска – вычеркивание из *D*-системы *всех* строк и столбцов без образования пустых строк. Другими словами, результирующее состояние в этом случае будет характеризоваться только совокупностью непустых усеченных доменов.

Описанные преобразования сохраняют множество решений *D*-системы. В некоторых случаях, например, при решении задачи “Выполнимость КНФ”, можно предложить дополнительные правила редукции, не сохраняющие множество корней, а сохраняющие лишь свойство выполнимости КНФ. Эти дополнительные правила неприменимы, если переменные, кроме прочего, связаны нелогическими ограничениями, диктуемыми спецификой предметной области, так как на некоторой итерации можно исключить все допустимые решения. Однако именно эти дополнительные ограничения, не позволяющие применять одни правила редукции, сами могут использоваться для ускорения процесса поиска, о чем пойдет речь несколько позже. Приведем пример, иллюстрирующий предложенные методы поиска.

Пример 1. Пусть имеется граф (Рис. 1) и требуется раскрасить его в три цвета (r – красный, g –зеленый, b – синий), чтобы смежные вершины были бы раскрашены в разные цвета.

Решение. Сопоставим каждой вершине графа соответствующий атрибут: $A - X_A$, $B - X_B$, $C - X_C$, $D - X_D$, $E - X_E$. Областью определения каждого атрибута является множество цветов: $\{r, g, b\}$. Запишем ограничения задачи в виде D-системы.

Сначала сформулируем ограничение, утверждающее, что пара вершин, например A и B, должна быть раскрашена в разные цвета. Это ограничение отрицает утверждение “Вершины A и B раскрашены в один цвет”, которое можно формализовать в виде следующей C-системы:

$$K1[X_AX_B] = \begin{bmatrix} \{r\} & \{r\} \\ \{g\} & \{g\} \\ \{b\} & \{b\} \end{bmatrix}.$$

А дополнение этой C-системы представляет собой искомую формализацию:

$$R1[X_AX_B] = \begin{bmatrix} \{g,b\} & \{g,b\} \\ \{r,b\} & \{r,b\} \\ \{g,r\} & \{g,r\} \end{bmatrix}.$$

По аналогии формируются ограничения: $R2[X_AX_C]$, $R3[X_AX_D]$, $R4[X_BX_C]$, $R5[X_CX_D]$. В пространстве атрибутов X_A, X_B, X_C, X_D, X_E запишем $R = R1 \cap_G \ldots \cap_G R5$:

	X_A	X_B	X_C	X_D	X_E
	$\{r,g,b\}$	$\{r,g,b\}$	$\{r,g,b\}$	$\{r,g,b\}$	$\{r,g,b\}$
1	$\{g,b\}$	$\{g,b\}$	$\varnothing$	$\varnothing$	$\varnothing$
2	$\{r,g\}$	$\{r,g\}$	$\varnothing$	$\varnothing$	$\varnothing$
3	$\{r,b\}$	$\{r,b\}$	$\varnothing$	$\varnothing$	$\varnothing$
4	$\{g,b\}$	$\varnothing$	$\{g,b\}$	$\varnothing$	$\varnothing$
5	$\{r,g\}$	$\varnothing$	$\{r,g\}$	$\varnothing$	$\varnothing$
6	$\{r,b\}$	$\varnothing$	$\{r,b\}$	$\varnothing$	$\varnothing$
7	$\{g,b\}$	$\varnothing$	$\varnothing$	$\{g,b\}$	$\varnothing$
8	$\{r,g\}$	$\varnothing$	$\varnothing$	$\{r,g\}$	$\varnothing$
9	$\{r,b\}$	$\varnothing$	$\varnothing$	$\{r,b\}$	$\varnothing$
10	$\varnothing$	$\{g,b\}$	$\{g,b\}$	$\varnothing$	$\varnothing$
11	$\varnothing$	$\{r,g\}$	$\{r,g\}$	$\varnothing$	$\varnothing$
12	$\varnothing$	$\{r,b\}$	$\{r,b\}$	$\varnothing$	$\varnothing$
13	$\varnothing$	$\varnothing$	$\{g,b\}$	$\{g,b\}$	$\varnothing$
14	$\varnothing$	$\varnothing$	$\{r,g\}$	$\{r,g\}$	$\varnothing$
15	$\varnothing$	$\varnothing$	$\{r,b\}$	$\{r,b\}$	$\varnothing$

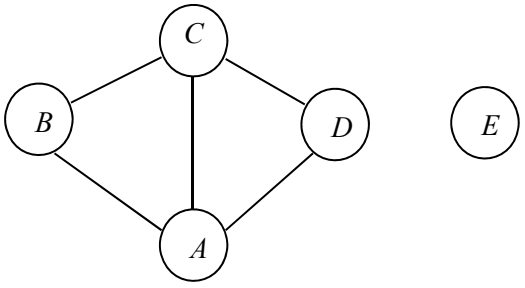

Рис. 1. Задача о раскраске графа

Видно, что атрибуту X_E соответствует столбец, содержащий лишь пустые компоненты, а это значит, что можно исключить атрибут X_E из дальнейшего рассмотрения, положив, например, что домен X_E равен $\{r\}$. Получим остаток:

	X_A	X_B	X_C	X_D
	$\{r,g,b\}$	$\{r,g,b\}$	$\{r,g,b\}$	$\{r,g,b\}$
1	$\{g,b\}$	$\{g,b\}$	$\varnothing$	$\varnothing$
2	$\{r,g\}$	$\{r,g\}$	$\varnothing$	$\varnothing$
3	$\{r,b\}$	$\{r,b\}$	$\varnothing$	$\varnothing$
4	$\{g,b\}$	$\varnothing$	$\{g,b\}$	$\varnothing$
5	$\{r,g\}$	$\varnothing$	$\{r,g\}$	$\varnothing$
6	$\{r,b\}$	$\varnothing$	$\{r,b\}$	$\varnothing$
7	$\{g,b\}$	$\varnothing$	$\varnothing$	$\{g,b\}$
8	$\{r,g\}$	$\varnothing$	$\varnothing$	$\{r,g\}$
9	$\{r,b\}$	$\varnothing$	$\varnothing$	$\{r,b\}$
10	$\varnothing$	$\{g,b\}$	$\{g,b\}$	$\varnothing$
11	$\varnothing$	$\{r,g\}$	$\{r,g\}$	$\varnothing$
12	$\varnothing$	$\{r,b\}$	$\{r,b\}$	$\varnothing$
13	$\varnothing$	$\varnothing$	$\{g,b\}$	$\{g,b\}$
14	$\varnothing$	$\varnothing$	$\{r,g\}$	$\{r,g\}$
15	$\varnothing$	$\varnothing$	$\{r,b\}$	$\{r,b\}$

Очевидно, что применение к только что полученной D-системе эвристики **Э1** ничего не даст. Пользуясь эвристикой **Э2**, выбираем атрибут (атрибуты), имеющий максимально возможное количество компонент в столбце D-системы. В нашем случае таких атрибутов два – X_A и X_C. Между этими вариантами выбор производится случайным образом. Допустим, выбрали атрибут X_A. Опираясь на **Э3**, ищем наиболее часто встречаемое значение в компонентах столбца. Если частота встречаемости значений в столбце одинакова, как в нашем случае, то берем первое попавшееся значение,

например r. Далее фактически происходит "настройка" D-системы на новый одноэлементный домен $\{r\}$.

Эта "настройка" сводится к исключению из компонент значений, отличных от r по правилу **У5**. В результате некоторые компоненты в атрибуте X_A становятся пустыми, а оставшиеся – полными, т.е. равными новому домену (в нашем случае $\{r\}$). Далее, используя **У4**, удаляем строки, содержащие полные компоненты – строки с номерами 2, 3, 5, 6, 8, 9.

Атрибут X_A, домен которого стал одноэлементным множеством $\{r\}$, мы можем исключить из дальнейшего рассмотрения, тогда получим такую D-систему:

	X_B	X_C	X_D
	$\{r,g,b\}$	$\{r,g,b\}$	$\{r,g,b\}$
1	$\{g,b\}$	$\varnothing$	$\varnothing$
4	$\varnothing$	$\{g,b\}$	$\varnothing$
7	$\varnothing$	$\varnothing$	$\{g,b\}$
10	$\{g,b\}$	$\{g,b\}$	$\varnothing$
11	$\{r,g\}$	$\{r,g\}$	$\varnothing$
12	$\{r,b\}$	$\{r,b\}$	$\varnothing$
13	$\varnothing$	$\{g,b\}$	$\{g,b\}$
14	$\varnothing$	$\{r,g\}$	$\{r,g\}$
15	$\varnothing$	$\{r,b\}$	$\{r,b\}$

Строки 1, 4, 7 оставшейся D-системы содержат лишь по одной непустой компоненте. Следовательно, по **У3** области определения атрибутов X_B, X_C, X_D могут быть усечены. Все эти области определения становятся равны множеству $\{g, b\}$.

Далее производится "настройка" полученной на предыдущем шаге D-системы уже на новые домены. По **У5** убираем лишние значения из компонент. По **У4** элиминируем строки 1, 4, 7, 10, 13:

	X_B	X_C	X_D
	$\{g,b\}$	$\{g,b\}$	$\{g,b\}$
11	$\{g\}$	$\{g\}$	$\varnothing$
12	$\{b\}$	$\{b\}$	$\varnothing$
14	$\varnothing$	$\{g\}$	$\{g\}$
15	$\varnothing$	$\{b\}$	$\{b\}$

Более редуцировать пространство поиска, не прибегая к ветвлению, не представляется возможным. Пробуем осуществить выбор ветки дерева поиска, опираясь на введенные эвристики.

Пользуясь **Э2**, рассматриваем атрибут X_C. Пусть для X_C выбрано значение g, тогда уходят строки 11 и 14, а сам атрибут X_C может быть исключен из рассмотрения. Имеем:

	X_B	X_D
	$\{g,b\}$	$\{g,b\}$
12	$\{b\}$	$\varnothing$
15	$\varnothing$	$\{b\}$

На основе правила **У3** можно сузить домены атрибутов X_B и X_D до одноэлементного множества $\{b\}$. Таким образом, получаем одно из возможных решений данной задачи, которое записывается в виде совокупности пар "атрибут – домен атрибута":

$$X_A - \{r\}; X_B - \{b\}; X_C - \{g\}; X_D - \{b\}; X_E - \{r\}.$$

Это означает, что вершины A и E можно раскрасить в красный цвет, вершины B и D – в синий, а вершину C – в зеленый цвет.

3. Обработка специальных ограничений

В реальных задачах часто встречаются типы ограничений, которые могут обрабатываться с помощью алгоритмов специального назначения, более эффективных по сравнению с методами общего назначения, рассматриваемыми до сих пор.

Ряд исследований по удовлетворению ограничений с конечными доменами использует понятие *глобального ограничения* [11]. Несмотря на то, что глобальные ограничения являются важным аспектом теории CSP, четкого их определения нет. Классическим примером глобального ограничения служит ограничение *Alldiff*, которое указывает, что все участвующие в нем переменные должны иметь разные значения. Это ограничение часто используется на практике и в связи с этим предлагается как встроенное ограничение во многих основных коммерческих и исследовательских системах программирования в ограничениях [12, 13].

У глобальных ограничений есть два преимущества. Во-первых, они облегчают задачу моделирования прикладной проблемы в виде задачи CSP. Вместо записи задачи с помощью многих исходных бинарных ограничений, мы описываем ее с помощью небольшого числа

глобальных ограничений для множеств переменных. Во-вторых, можно разработать алгоритм распространения ограничений специального вида, учитывающий особенности ограничения и поэтому намного более эффективный.

Так, например, ограничение *Alldiff* может обрабатываться, как и любое другое ограничение, если *Alldiff* записать в виде строк *D*-системы. Этот способ порождает довольно громоздкие конструкции. Альтернативный способ – разработать специальную процедуру, дополняющую изложенные ранее базовые методы, которая бы гарантировала соблюдение ограничения *Alldiff* на уровне доменов переменных.

Другими примерами глобальных ограничений являются глобальное *кардинальное ограничение* (global cardinality constraint (gcc)) [14] и *кумулятивное ограничение* [12]. Глобальное кардинальное ограничение определяет условие, что число переменных, которым присвоены значения, должно быть между заданными верхней и нижней границами. Эти границы могут быть различны для каждого значения. Кумулятивное ограничение над множеством переменных, представляющих времена выполнения различных работ, гарантирует, что работы упорядочены с учетом имеющихся в наличии величин ресурсов, используемых в любые периоды времени. Ограничения обоих указанных типов обычно используются при составлении расписаний, упорядочении работ и в календарном планировании.

Покажем на конкретном примере, каким образом осуществляется совместная обработка логических и кумулятивных ограничений в рамках предлагаемого подхода.

***Пример* 2.** Пусть имеется список заданий (работ) и ограничения на то, какие задачи могут быть выполнены одновременно, выполнение каких заданий должно предшествовать выполнению других и т. д. Для решения этой задачи нужно найти назначение времен начал работ заданиям так, чтобы все ограничения удовлетворялись. Рассмотрим задачу составления расписания для пяти заданий T_1, T_2, T_3, T_4, T_5, каждое из которых может быть выполнено за один час. Задания могут начинаться в 1:00, 2:00 или 3:00. Любые работы могут выполняться одновременно, учитывая ограничения на то, что: 1) T_1 начинается после T_3, 2) T_3 начинается до T_4 и после T_5, 3) T_2 не может начинаться в то же время, что T_1 или T_4, 4) T_4 не может начинаться в 2:00.

Кроме чисто логических ограничений, накладывается следующее кумулятивное ограничение: для выполнения T_1 требуется 5 человек, для задания T_2 – 7 человек, для T_3 – 3 человека, для T_4 – 5 человек, для T_5 – 4 человека. Количество работ, которые могут выполняться одновременно, ограничено имеющейся рабочей силой (10 человек).

Решение. Можно построить модель составления графика, введя, пять атрибутов X_1, X_2, X_3, X_4, X_5, соответствующих заданиям, с доменами $\{1, 2, 3\}$, где: 1 – 1:00, 2 – 2:00, 3 – 3:00.

Видно, что формулировка задачи опирается на два типа бинарных отношений: “начинаться до/позже чем” и “выполняться одновременно/не одновременно”. Запишем условия задачи в виде рассмотренных ранее матрицеподобных структур.

Введем атрибуты X_1, X_2, X_3, X_4, X_5, которые соотносятся с заданиями T_1, T_2, T_3, T_4, T_5, соответственно. Все атрибуты имеют одну и ту же область определения $D = \{1, 2, 3\}$ – времена начала работ. Рассмотрим, как формализовать в виде *D*-системы отношение “T_1 начинается после T_3”. Отношение “T_1 начинается до или одновременно с T_3” на языке АК выглядит так:

$$C1[X_1X_3] = \begin{bmatrix} \{1\} & * \\ \{2\} & \{2,3\} \\ \{3\} & \{3\} \end{bmatrix}.$$

Тогда его дополнение, представляющее собой требуемую формализацию, записывается в виде:

$$D1[X_1X_3] = \begin{bmatrix} \{2,3\} & \varnothing \\ \{1,3\} & \{1\} \\ \{1,2\} & \{1,2\} \end{bmatrix}.$$

По аналогии, сформируем *D*-системы *D*2 и *D*3, соответствующие условиям “T_3 начинается после T_5”, “T_4 начинается после T_3” (равносильно утверждению “T_3 начинается до T_4”, заданному в условии задачи):

$$D2[X_3X_5] = \begin{bmatrix} \{2,3\} & \varnothing \\ \{1,3\} & \{1\} \\ \{1,2\} & \{1,2\} \end{bmatrix}, \; D3[X_4X_3] = \begin{bmatrix} \{2,3\} & \varnothing \\ \{1,3\} & \{1\} \\ \{1,2\} & \{1,2\} \end{bmatrix}.$$

Формализация следующего отношения “T_2 не может начинаться в то же время, что T_1 или T_4” осуществляется путем пересечения двух представленных ниже *D*-систем *D*4 и. *D*5.

$$D4[X_1X_2] = \begin{bmatrix} \{2,3\} & \{2,3\} \\ \{1,3\} & \{1,3\} \\ \{1,2\} & \{1,2\} \end{bmatrix}, D5[X_2X_4] = \begin{bmatrix} \{2,3\} & \{2,3\} \\ \{1,3\} & \{1,3\} \\ \{1,2\} & \{1,2\} \end{bmatrix}.$$

Так $D4[X_1X_2]$ получается вычислением дополнения от

$$C4 = \begin{bmatrix} \{1\} & \{1\} \\ \{2\} & \{2\} \\ \{3\} & \{3\} \end{bmatrix},$$

моделирующей утверждение "T_2 начинается в то же время, что T_1".

Наконец, утверждение "T_4 не может начинаться в 2:00" записывается как отрицание утверждения "T_4 начинается в 2:00" (запишем его в полном пространстве признаков):

$$D6[X_1X_2X_3X_4X_5] = \overline{[*\ \ *\ \ *\ \ \{2\}\ \ *]} =$$

$$=]\varnothing\ \ \varnothing\ \ \varnothing\ \ \{1,3\}\ \ \varnothing[.$$

Тогда общая формулировка задачи может быть выражена как следующая D-система:

	X_1	X_2	X_3	X_4	X_5
	$\{1,2,3\}$	$\{1,2,3\}$	$\{1,2,3\}$	$\{1,2,3\}$	$\{1,2,3\}$
1	$\{2,3\}$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
2	$\{1,3\}$	$\varnothing$	$\{1\}$	$\varnothing$	$\varnothing$
3	$\{1,2\}$	$\varnothing$	$\{1,2\}$	$\varnothing$	$\varnothing$
4	$\varnothing$	$\varnothing$	$\{2,3\}$	$\varnothing$	$\varnothing$
5	$\varnothing$	$\varnothing$	$\{1,3\}$	$\varnothing$	$\{1\}$
6	$\varnothing$	$\varnothing$	$\{1,2\}$	$\varnothing$	$\{1,2\}$
7	$\varnothing$	$\varnothing$	$\varnothing$	$\{2,3\}$	$\varnothing$
8	$\varnothing$	$\varnothing$	$\{1\}$	$\{1,3\}$	$\varnothing$
9	$\varnothing$	$\varnothing$	$\{1,2\}$	$\{1,2\}$	$\varnothing$
10	$\{2,3\}$	$\{2,3\}$	$\varnothing$	$\varnothing$	$\varnothing$
11	$\{1,3\}$	$\{1,3\}$	$\varnothing$	$\varnothing$	$\varnothing$
12	$\{1,2\}$	$\{1,2\}$	$\varnothing$	$\varnothing$	$\varnothing$
13	$\varnothing$	$\{2,3\}$	$\varnothing$	$\{2,3\}$	$\varnothing$
14	$\varnothing$	$\{1,3\}$	$\varnothing$	$\{1,3\}$	$\varnothing$
15	$\varnothing$	$\{1,2\}$	$\varnothing$	$\{1,2\}$	$\varnothing$
16	$\varnothing$	$\varnothing$	$\varnothing$	$\{1,3\}$	$\varnothing$

Анализируя строки 1, 4, 7, 16, по **У3** можно усечь домены для X_1, X_3, X_4, получим такие соответствия между атрибутами и их доменами: X_1 – {2, 3}, X_2 – {1, 2, 3}, X_3 – {2, 3}, X_4 – {3}, X_5 – {1, 2, 3}. При этом для одновременного выполнения T_4 и T_1 требуется 10 человек, T_4 и T_2 – 12 человек, T_4 и T_3 – 8 человек, а для выполнения работ T_4 и T_5 – 9 человек. Следовательно, кумулятивное ограничение для момента времени "3:00" нарушено лишь для заданий T_4 и T_2. Для устранения этой некорректности из домена атрибута X_2 исключается значение "3", имеем:

X_1 – {2, 3}, X_2 – {1, 2}, X_3 – {2, 3}, X_4 – {3}, X_5 – {1, 2, 3}.

Затем атрибут X_4 элиминируется из D-исходной системы, поскольку его домен стал одноэлементным множеством {3}. В результате "настройки" D-системы на новые домены, применяя правило **У4**, исключаются из рассмотрения строки 1, 4, 7, 8, 10, 12, 13, 14, 15, 16.

Далее, используя **У5**, получим (здесь и далее, помимо самой D-системы, будем записывать элиминированные атрибуты с их доменами, т.е. давать полное описание состояния CSP):

X_4 – {3}

	X_1	X_2	X_3	X_5
	$\{2,3\}$	$\{1,2\}$	$\{2,3\}$	$\{1,2,3\}$
2	$\{3\}$	$\varnothing$	$\varnothing$	$\varnothing$
3	$\{2\}$	$\varnothing$	$\{2\}$	$\varnothing$
5	$\varnothing$	$\varnothing$	$\{3\}$	$\{1\}$
6	$\varnothing$	$\varnothing$	$\{2\}$	$\{1,2\}$
9	$\varnothing$	$\varnothing$	$\{2\}$	$\varnothing$
11	$\{3\}$	$\varnothing$	$\varnothing$	$\varnothing$

На основе анализа строки 2 можно сузить домен атрибута X_1 до единственного значения (X_1 – {3}). Поскольку ранее было получено X_4 – {3}, то с учетом ограничений на рабочую силу суммарное количество персонала, задействованного для работ в T_1 и T_4 в 3:00, равно 9. Значит, никакая другая работа уже не может быть выполнена одновременно с T_1 и T_4. Другими словами, значение "3" можно исключить из областей определения других атрибутов. На текущем шаге имеем: X_1 – {3}, X_2 – {1,2}, X_3 – {2}, X_5 – {1,2}. Поскольку более не обнаружено нарушения кумулятивного ограничения и нельзя усечь домены, то перейдем к рассмотрению D-системы, оставшейся после "настройки" на новые домены (согласно **У4** и **У5**, уходят все строчки, кроме пятой, а в пятой остается лишь

одна непустая компонента). Атрибуты X_1 и X_3 элиминируются из D-системы. Получим:

$$X_1 - \{3\}, X_3 - \{2\}, X_4 - \{3\}$$

X_2	X_5
{1,2}	{1,2}
]∅	{1}[

Откуда X_5={1}, следовательно, X_2={2}, поскольку работы T_2 и T_5 не могут выполняться одновременно. В результате, нами получено решение:

$$X_1 - \{3\}, X_2 - \{2\}, X_3 - \{2\}, X_4 - \{3\}, X_5 - \{1\}.$$

Другими словами, в 1:00 выполняется T_5, в 2:00 выполняется одновременно T_2 и T_3, в 3:00 одновременно производятся работы T_1 и T_4.

Заключение

Разработаны новые методы решения задач CSP, использующие матричное представление конечных предикатов. Они позволяют обойтись без разложения логических ограничений в бинарные отношения. Анализ внутренней структуры таких матриц дает возможность эффективно редуцировать пространство поиска. Впервые в АК рассмотрена инкрементная постановка задачи поиска и разработан метод распространения ограничений, основанный на пошаговом усечении доменов атрибутов и получении решения задачи CSP в виде совокупности доменов. В отличие от имевшихся в АК методов “слепого” поиска, в разработанных методах широко применяются эвристики. В рамках предлагаемого подхода базовые методы могут быть легко адаптированы для обработки специальных ограничений, что обеспечивает гибкость настройки на конкретную предметную область.

Литература

1. Щербина,О. А. Удовлетворение ограничений и программирование в ограничениях / О. А. Щербина // препринт, 2012.
2. Рассел С., Норвиг П. Искусственный интеллект: современный подход. 2-е изд. / пер. с англ.; ред. К.А. Птицына. – М.: Изд. дом «Вильямс», 2006. 1408 с.
3. Bartak R. Constraint Programming: In Pursuit of the Holy Grail // Proceedings of the Week of Doctoral Students (WDS99), Part IV. Prague: MatFyzPress, 1999. P. 555–564.
4. Ruttkay Zs. Constraint satisfaction a survey // CWI Quarterly.1998. V. 11. P. 163–214.
5. Кулик Б.А., Зуенко А.А., Фридман А.Я. Алгебраический подход к интеллектуальной обработке данных и знаний. – СПб.: Изд-во Политехн. ун-та, 2010, 235 с.
6. Зуенко, А.А. Матрицеподобные вычисления в задачах удовлетворения ограничений /А.А. Зуенко // Шестая Всероссийская мультиконференция по проблемам управления (30 сентября – 5 октября 2013 г.). Материалы мультиконференции: в 4-х т. – Ростов-на-Дону: Издательство Южного федерального университета, 2013. Т.1. – С. 30-34.
7. Зуенко А.А. Унификация обработки данных и знаний на основе общей теории многоместных отношений / А.А. Зуенко, Б.А. Кулик, А.Я. Фридман // Искусственный интеллект и принятие решений, 2010. Вып. 3. – С.52-62.
8. Кулик, Б.А. Дедуктивные и пересматриваемые рассуждения на основе единого алгебраического подхода / Б.А. Кулик, А.А. Зуенко, А.Я. Фридман // Искусственный интеллект и принятие решений, 2013. Вып. 4. – С.34-44.
9. Kulik Boris, Fridman Alexander, Zuenko Alexander. Logical Inference and Defeasible Reasoning in N-tuple Algebra. In: “Diagnostic Test Approaches to Machine Learning and Commonsense Reasoning Systems”, IGI Global, P 102-128.
10. Zakrevskij Arkadij. Integrated Model of Inductive-Deductive Inference Based on Finite Predicates and Implicati ve Regularities. In: “Diagnostic Test Approaches to Machine Learning and Commonsense Reasoning Systems”, IGI Global, P 1-12.
11. Rossi F., van Beek P., Walsh T. Chapter 4 «Constraint Programming» in Foundations of Artificial Intelligence. // Handbook of Knowledge Representation / F. van Harmelen, V. Lifschitz, B. Porter (eds.) 2008. V. 3. P. 181–211.
12. Aggoun A. Beldiceanu N. Extending CHIP in order to solve complex scheduling and placement problems // Mathl. Comput. Modelling. 1993. V. 17. P. 57–73.
13. Beldiceanu N. Global constraints as graph properties on structured network of elementary constraints of the same type. Technical Report T2000/01, SICS, 2000.
14. R´egin J. Generalized arc consistency for global cardinality constraint // Proceedings of the Thirteenth National Conference on Artificial Intelligence. Portland, 1996. P. 209–215.

Зуенко Александр Анатольевич. Научный сотрудник Института информатики и математического моделирования технологических процессов КолНЦ РАН, доцент Кольского филиала Петрозаводского государственного университета. В 2005 г. окончил Петрозаводский государственный университет. Кандидат технических наук. Автор 90 печатных работ, в том числе трех монографий. Область научных интересов: системы моделирования на основе открытой концептуальной модели предметной области, представление и обработка знаний. E-mail: zuenko@iimm.ru