Теоретические и экспериментальные оценки сложности методов локального распространения в задачах программирования в ограничениях

Семенов В.А., Сидяка О.В.

Аннотация. Обсуждаются вопросы универсальности и эффективности методов локального распространения значений и степеней свободы применительно к задачам программирования в ограничениях. На основе сравнительного анализа обозначаются границы применимости методов и даются теоретические оценки их сложности. Отмечается важность построения и использования комбинированных алгоритмов, обеспечивающих надежное решение широких классов задач за полиномиальное время. Для предложенного комбинированного алгоритма проводятся серии вычислительных экспериментов, моделирующих системы ограничений переменной размерности с разным характером зависимостей по данным. Обсуждаются полученные экспериментальные оценки сложности алгоритма и отмечаются его конкурентные преимущества над традиционными методами локального распространения.

1. Введение

В последние годы логическое программирование в ограничениях CLP (Constraint Logic Programming) [1] и лежащие в его основе подходы к разрешению систем ограничений CSP (Constraint Satisfaction Problem) [2] получили заметное развитие. Технологии и методы программирования в ограничениях успешно применяются:

— при организации “интеллектуальных” графических интерфейсов пользователя, предусматривающих специальные правила для эргономичного расположения оконных элементов на экране [3, 4];

— в активных базах данных для определения полных (удовлетворяющих все условия целостности) и корректных (исключающих зацикливание) политик каскадного обновления семантически связанных элементов данных [5];

— в системах CAD/CAM/CAE для решения разнообразных инженерных задач, требующих автоматического и согласованного

пересчета параметров модели в зависимости от решений пользователя, принимаемых в ходе рабочей сессии [6, 7].

Парадигма программирования в ограничениях, основанная на непосредственной интерпретации прикладной задачи в терминах переменных, множеств допустимых значений и систем ограничений, оказалась привлекательной и конструктивной для построения сложных вычислительных приложений, в процессе выполнения которых возникает необходимость динамической идентификации и решения разнообразных математических задач алгебраического типа. Данная парадигма предполагает, что программист описывает прикладную задачу путем декларирования переменных и отношений между ними и полностью освобождается от написания императивной части программного приложения, ответственной за решение задачи с использованием тех или иных алгоритмов. Поскольку описываемые формальным образом ограничения могут иметь вид уравнений, неравенств или логических выражений самого общего вида, анализ зависимостей между переменными, идентификация типа математической задачи, а также ее непосредственное решение требуют целого арсенала математических методов и программных средств.

Для разрешения ограничений обычно применяются альтернативные подходы.

Численный подход [8] использует предварительную редукцию исходной системы ограничений к системе нелинейных алгебраических уравнений или к задаче условной нелинейной оптимизации. В зависимости от характера зависимостей задачи могут решаться разнообразными прямыми и итерационными методами, в частности, методами квазиньютоновского типа, методами сопряженных градиентов, методами матричной факторизации, симплекс-методом. Невысокая эффективность подхода объясняется проблемами локальной и глобальной сходимости итерационных методов для нелинейных систем ограничений, а также высокой сложностью прямых методов при решении линейных задач. Вместе с тем, при наличии сложных зависимостей между переменными данный подход часто оказывается единственно возможным.

Символьный [9] подход предполагает те же редукционные схемы, однако математические задачи решаются методами, основанными на символьных преобразованиях. В случае нелинейных ограничений общего вида время символьного решения растет экспоненциально с ростом числа ограничений. При этом шансы на успешное разрешение всех ограничений обычно невысоки. Для систем специального вида символьные методы могут дать заметный выигрыш.

Кластеризация [10] подразумевает декомпозицию исходной системы ограничений на подсистемы таким образом, чтобы обеспечить разрешимость каждой из них относительно внутренних переменных. Тогда последовательным обходом подсистем можно попытаться удовлетворить все ограничения исходной системы. При наличии изолированных групп

переменных кластеризация существенно упрощает и ускоряет процесс решения. В ряде случаев ее целесообразно применять в сочетании с методами анализа и идентификации типов математических задач, связанных с выделенными подсистемами ограничений.

Методы *локального распространения* [11–13] составляют наиболее распространенный и перспективный подход к решению больших систем ограничений, для которых известны или могут быть выведены альтернативные правила явного разрешения одних переменных относительно других. Как правило, современные реализации методов предусматривают предварительный этап планирования решения, на котором определяется порядок и способ разрешения ограничений, и этап непосредственного решения, на котором план, представленный в виде последовательности правил, применяется для расчета неизвестных переменных. Единожды построенный план может многократно использоваться для решения подобных задач в инкрементальной постановке, допускающей частичное изменение системы ограничений и их параметров.

Различают методы локального распространения значений и степеней свободы.

В методе локального распространения значений часть переменных принимается в качестве параметров задачи, через которые могут быть последовательно выражены остальные неизвестные. Пусть, например, задана система ограничений $x_1 + x_2 = x_3$ и $x_1 + x_2 + x_3 = x_4$, и переменные x_1 и x_3 выбраны в качестве параметров задачи. Тогда остальные неизвестные могут быть найдены путем последовательного применения правил $x_2 = x_3 - x_1$ и $x_4 = x_1 + x_2 + x_3$.

В методе локального распространения степеней свободы [11] происходит последовательный поиск и исключение групп свободных переменных и ассоциированных с ними ограничений. Свободные переменные — переменные, входящие лишь в одно ограничение исходной системы и участвующие в качестве выходов одного из его решающих правил. Найденное правило добавляется в начало плана решения, а процесс продолжается до тех пор, пока не исчерпаны все ограничения исходной системы и остаются свободные переменные.

В разделе 2 мы рассмотрим формальную постановку задачи в ограничениях, разрешимую для методов локального распространения, и обоснуем ее оценки сложности. В разделе 3 приведем алгоритмические версии методов локального распространения значений и степеней свободы и построим комбинированный алгоритм, сочетающий элементы двух методов. Раздел 4 посвящен представлению и анализу результатов вычислительных экспериментов, убедительно доказывающих преимущества комбинированного алгоритма над традиционными методами локального распространения.

2. Задачи в ограничениях

Задачи в ограничениях обычно описываются путем определения множества неизвестных переменных и алгебраических зависимостей между ними. При этом процесс решения заключается в локализации областей значений переменных или в поиске значений, удовлетворяющих заданным зависимостям. Будем рассматривать только *ограничения потока данных*. Подобные ограничения могут разрешаться на основе предварительно выведенных или заданных правил. Каждое такое правило предоставляет метод разрешения ограничения с сигнатурой, определяющей, какие переменные задачи являются его входными параметрами, а какие — выходными. Каждое правило при этом имеет хотя бы один выходной параметр, подлежащий модификации в результате применения правила и разрешения соответствующего ограничения.

2.1. Формальная постановка

Итак, пусть задано множество переменных $X = \{x_i \mid i = 1..n\}$ и система ограничений на нем $C = \{C_j(X_j) \mid j = 1..m, X_j \subseteq X\}$. Будем считать, что для каждого ограничения системы $C_j \in C$ известен соответствующий набор правил $R_j = \{R_{jk}(X_{jk}^{IN}, X_{jk}^{OUT}) \mid k = 1..k_j, X_{jk}^{IN} \subset X_j, X_{jk}^{OUT} \subseteq X_j, X_{jk}^{IN} \cap X_{jk}^{OUT} = \varnothing\}$ разрешения относительно переменных, участвующих в них в качестве входных и выходных параметров.

Задача стоит в нахождении плана решения в виде последовательности правил $S : \{1..\mathrm{m}\} \to \mathop{\mathrm{Y}}_{j=1}^{m}\left(\mathop{\mathrm{Y}}_{k=1}^{k_j} R_{jk}\right)$, удовлетворяющей следующим условиям:

1. последовательность правил полна для разрешения всех ограничений исходной системы $\forall i \in \{1..m\} \exists j \in \{1..m\} : S(j) \in R_i$;
2. последовательность правил неконфликтна и исключает случаи повторной модификации установленных переменных и нарушения разрешенных ограничений системы. Формально данное условие выражается как

$$\forall i', i'' \in \{1..m\}, i' < i'' : \begin{cases} X_{j'k'}^{OUT} \cap X_{j''k''}^{OUT} = \varnothing \\ X_{j'k'}^{IN} \cap X_{j''k''}^{OUT} = \varnothing \end{cases} \quad \text{ãääå} \quad \mathrm{S}(i') = \mathrm{R}_{\mathrm{j'k'}}, \quad \mathrm{S}(i'') = \mathrm{R}_{\mathrm{j''k''}}$$

Задача предыдущего раздела формально представляется как система ограничений относительно множества неизвестных переменных $X = \{x_1, x_2, x_3, x_4\}$:

$$\begin{cases} C_1 : x_1 + x_2 = x_3 \\ C_2 : x_1 + x_2 + x_3 = x_4 \end{cases}$$

.Необходимые решающие правила могут быть заданы, например, следующим образом:

$$\begin{cases} R_{11}: x_1 = x_3 - x_2;\ R_{12}: x_2 = x_3 - x_1;\ R_{13}: x_3 = x_1 + x_2 \\ R_{21}: x_1 = x_4 - x_2 - x_3;\ R_{22}: x_2 = x_4 - x_1 - x_3;\ R_{23}: x_3 = x_4 - x_1 - x_2;\ R_{24}: x_4 = x_1 + x_2 + x_3 \end{cases}$$

Планом решения данной задачи является $S = \{R_{13}, R_{24}\}$. Можно заметить, что решение не единственно. Например, $S = \{R_{11}, R_{24}\}$ также является планом ее решения.

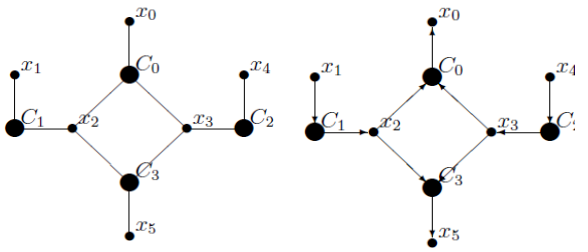
2.2. Визуальное представление задачи и плана решения

Естественным визуальным представлением задачи в ограничениях является двудольный ненаправленный граф $G(X,C,R)$ [2-5], где X – множество вершин, ассоциированных с переменными задачи, C – множество вершин, ассоциированных с ограничениями задачи (на рисунках ниже отображены большими кругами), и R – множество ненаправленных ребер графа, соединяющих вершины ограничений с соответствующими вершинами переменных. Переменная задачи, участвующая в некотором ограничении, отображается ребром, соединяющим соответствующие вершины двудольного графа.

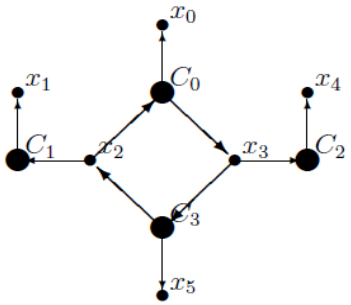
Результатом решения задачи в ограничениях является план в виде последовательности правил, каждое из которых приводит к удовлетворению одного из ограничений исходной системы. План решения визуально отображается на графе с помощью направленных ребер. Комбинация ориентаций ребер, инцидентных вершине некоторого ограничения, определяет для него сигнатуру решающего правила. Входные параметры правила отображаются на графе ребрами, входящими в вершину ограничения, а выходные — ребрами, выходящими из нее. Ребра переменных, участвующих в ограничении, но не задействованных в выбранном правиле его разрешения, исключаются из представления графа. Двудольный ориентированный граф плана решения исходной задачи в ограничениях, представимой в виде $G(X,C,R)$, обозначим как $\overline{G}(X,C,R)$.

Принцип разрешимости задач в ограничениях также имеет прозрачную визуальную интерпретацию. Задача разрешима, если план ее решения, полученный в виде $\overline{G}(X,C,R)$, не содержит циклов и каждая его вершина, ассоциированная с переменной задачи, имеет не более одного входящего ребра. Циклом в данном случае называется упорядоченный набор пар вершин $x_{n_1}, C_{n_2}, x_{n_3}, C_{n_4}, ..., x_{n_{k-1}}, C_{n_k} \in \overline{G}$ такой, что вершины-переменные $x_{n_1}, x_{n_3}, ..., x_{n_{k-1}} \in X$ и вершины-ограничения $C_{n_2}, C_{n_4}, ..., C_{n_k} \in C$ соединены последовательно направленными ребрами от вершины x_{n_i} к вершине $C_{n_{i+1}}$ для

всех $i = 1..k-1$, от вершины C_{n_i} к вершине $x_{n_{i+1}}$ для всех $i = 2..k-2$, и от вершины C_{n_k} к вершине x_{n_1}.



(а) (б)



(в)

Данные условия гарантируют, что существует конечная неконфликтная последовательность правил, применяя которую, можно разрешить все ограничения исходной системы. Назовем задачу потенциально разрешимой, если она разрешима, однако существует, по крайней мере, один циклический план решения $\overline{G}(X,C,R)$. Для задачи, неразрешимой в обсуждаемой постановке, не может быть построен ациклический план решения $\overline{G}(X,C,R)$.

2.3. Вычислительная сложность задачи в ограничениях

Обсудим вопрос вычислительной сложности задач в ограничениях потока данных [14, 15]. Как правило, подобные задачи недоопределены, что приводит к существованию большого числа решений, а наличие альтернативных правил для каждого ограничения исходной системы обуславливает и

многовариантность способов их нахождения. Покажем, что задача в обсуждаемой общей постановке является NP-полной в силу того, что частные постановки также не разрешимы за полиномиальное время. С этой целью рассмотрим классическую задачу раскраски графа в k цветов и переформулируем ее в терминах эквивалентной задачи в ограничениях. Каждой вершине графа поставим в соответствие некоторую переменную задачи, означающую ее цвет, а каждому ребру — ограничение в виде условия несовпадения цветов, связанных с инцидентными вершинами. Каждое такое ограничение можно удовлетворить одним из $k(k-1)$ правил, каждое из которых перекрашивает две инцидентные вершины в разные цвета. Сформулированная задача в ограничениях имеет решение тогда и только тогда, когда исходный граф имеет раскраску в k цветов, что доказывает их эквивалентность и NP-полноту обсуждаемой задачи.

Известно, что в общем случае задача в ограничениях не решается эффективно, однако при переходе к частным упрощенным постановкам возникают альтернативные возможности, обеспечивающие поиск решения за полиномиальное время. В данной работе мы строим комбинированный алгоритм, который, с одной стороны, гарантирует решение в самой общей постановке, а с другой стороны — осуществляет поиск решения за полиномиальное время в частных, но распространенных на практике случаях.

3. Основные алгоритмы локального распространения

Прежде всего, приведем базовые алгоритмы для методов локального распространения значений и степеней свободы, на которых будет основываться предложенный комбинированный алгоритм.

3.1. Алгоритм распространения степеней свободы

Алгоритм распространения степеней свободы [11] гарантированно находит решение за полиномиальное время, если оно существует, и информирует об его отсутствии в противном случае. Однако постановка задачи имеет одно принципиальное уточнение — переменные каждого ограничения исходной системы должны участвовать во всех правилах его разрешения в качестве входных или выходных параметров. Данное условие выражается формальным образом: для любого ограничения системы $\forall C_j(X_j) \in C$, $j = 1..m$ и для любого его правила $\forall R_{jk}(X_{jk}^{IN}, X_{jk}^{OUT}) \in R_j$, $k = 1..k_j$ имеет место $X_{jk}^{IN} \cup X_{jk}^{OUT} = X_j$.

Ниже приведена поэтапная схема алгоритма.

1. Инициализируем план решения $S = \varnothing$.

2. Формируем подмножество всех свободных переменных задачи $X' \subseteq X$. Напомним, что свободной называется переменная, входящая лишь в одно ограничение исходной системы и участвующая в качестве выхода одного из его решающих правил. Формальное определение свободной переменной выглядит следующим образом:

$$x' \in X' \Leftrightarrow \begin{cases} \exists j' \in (1..m) : x' \in X_{j'}; \forall\ j \in (1..m), j \neq j' : x' \notin X_j \\ \exists k \in (1..k_{j'}) : R_{j'k} \in R_{j'}, \{x'\} = X_{j'k}^{OUT}) \end{cases}$$

3. Выбираем произвольную свободную переменную $x' \in X'$ и устанавливаем ассоциированное с ней ограничение $C_{j'}$, $\{x'\} = X_{j'}$ (в силу условий задачи оно всегда единственно).

4. Выбираем решающее правило $R_{j'k'}$ для ограничения $C_{j'}$ с фактическим выходным параметром-переменной $\{x'\} = X_{j'k'}^{OUT}$. Добавляем правило в начало плана решения S.

5. Корректируем представление задачи, удаляя переменную x' и связанное с ней ограничение $C_{j'}$. Обновляем подмножество свободных переменных $X' \subseteq X$ с учетом проведенных изменений.

6. Если $X' \neq \varnothing$, то переходим к пункту 3 и выбираем следующую свободную переменную.

7. Если множество ограничений в текущем представлении задачи исчерпано ($C = \varnothing$), то **план решения найден**. Если множество ограничений не исчерпано ($C \neq \varnothing$), то **плана решения не существует**.

Итак, имеет место следующее утверждение о корректности приведенного алгоритма.

Теорема. Алгоритм распространения степеней свободы завершает работу либо когда план решения найден, либо когда план не существует.

Доказательство.

□ Доказательство корректности алгоритма сводится к доказательству двух вспомогательных утверждений:

a. Если алгоритм завершил работу при исчерпании всех ограничений, то построенный план решения корректен.

b. Если алгоритм завершил работу при оставшихся ограничениях, то задача не разрешима.

Для доказательства первого утверждения перенумеруем все ограничения и переменные исходной системы сквозным образом в порядке, обратном исключению свободных переменных и ассоциированных с ними ограничений так, чтобы ограничения непосредственно предшествовали исключаемым переменным. Иными словами, перенумеруем их в соответствии с порядком применения правил в построенном плане решения. Тогда переменные, не являющиеся выходными параметрами ни одного из правил плана решения, окажутся в самом начале списка. При подобном расположении все ребра в графе плана имеют направление от элементов (переменных и ограничений), располагающихся выше по списку, к элементам (ограничениям и переменным соответственно) ниже по списку, что исключает циклы в графическом представлении плана. В самом деле, данному направлению соответствуют все ребра графа, а именно:

— ребра, соединяющие ограничения и выходные переменные правил, примененных для их разрешения;

— ребра, инцидентные переменным, которые не использовались в качестве выходов ни одного из решающих правил плана и, поэтому, оказались в самом начале списка;

— ребра, инцидентные переменным, которые использовались в качестве выходов одного из решающих правил плана, но были удалены в ходе его построения и поэтому не могли служить входными параметрами для правил разрешения ограничений, находящихся выше по списку.

Доказательство второго утверждения основывается на следующих рассуждениях относительно наличия циклов в графическом представлении плана. Во-первых, построенный без циклов план должен иметь хотя бы одно правило с выходными переменными, не используемыми в качестве входных параметров других правил. Если такого правила не существует, то путь в графе плана, проведенный от некоторого ограничения к выходным переменным и затем от них (которые по предположению существуют) к входным параметрам следующих ограничений, неизбежно замкнется. Во-вторых, в построенном графе без циклов не может быть вершины-переменной с несколькими входящими ребрами от разных ограничений. Ограничение, имеющее в качестве выхода переменную без выходных ребер, в качестве выходов имеет только свободные переменные. Поэтому, если в результирующем графе нет ограничения со свободными выходными переменными, то план решения не может быть представлен графом без циклов. ■

3.2. Алгоритм распространения значений, основанный на оценке перспективности выбираемых правил

Обсудим метод локального распространения значений и построим его алгоритмическую версию, использующую дополнительную спекулятивную оценку перспективности выбираемых правил. Подобная оценка позволяет выстраивать план решения локальным образом, но с учетом перспективы разрешения ограничений, анализируемых на последующих этапах алгоритма.

В основе оценки лежат следующие общие соображения. Во-первых, любые два правила плана не могут иметь общие выходные параметры. В противном случае ограничение, удовлетворенное первым, может быть нарушено при попытке разрешения второго ограничения и связанной с ней повторной модификации выходной переменной. Во-вторых, правила, использующие некоторые переменные в качестве входных, должны выполняться строго после правил, для которых эти переменные являются выходными параметрами. На каждом шаге обсуждаемого алгоритма проводится оценка зависимостей и выбирается то правило, применение которого гарантирует разрешимость любого из оставшихся ограничений. Тем самым, процесс планирования решения, реализуемый алгоритмом, всегда оказывается перспективным и может быть продолжен, по крайней мере, на последующем шаге разрешения очередного ограничения.

Ниже приводится пошаговая схема алгоритма, предполагающая последовательное разрешение ограничений задачи методом локального распространения значений. В случае успеха алгоритмом обрабатывается очередное ограничение, в противном случае — осуществляется возврат к предыдущему шагу и предпринимается попытка внести изменения в план решения. Алгоритм программно реализуется, используя аппарат рекурсивных функций.

1. Формируем упорядоченные множества разрешенных и неразрешенных ограничений задачи $C' = \varnothing$ и $C'' = C$ соответственно. Устанавливаем номер итерации $i = 0$ (совпадающий с количеством разрешенных ограничений в системе $i = |C'|$).

2. Инициализируем переменную цикла $j = 0$ для перебора неразрешенных ограничений из множества C''.

3. Устанавливаем индекс текущего обрабатываемого ограничения $j = j + 1$.

4. Если $j > |C''|$, то множество исчерпано и на данной итерации не удалось удовлетворить ни одно ограничение. В этом случае, если множество разрешенных ограничений C' пусто, то это означает,

что перебраны все варианты, **план решения не существует** и следует завершить работу. Если $C' \neq \varnothing$, то последнее разрешенное ограничение перемещается из множества C' обратно в C'' и осуществляется возврат к предыдущей итерации с номером $i = i - 1$, к пункту 7.

5. Пытаемся разрешить ограничение C''_j. Для этого формируем множество перспективных правил $R^*_j \subseteq R_j$. Перспективными считаются те правила, при включении которых в план решения, каждое из оставшихся ограничений C'' может быть удовлетворено, по крайней мере, с помощью одного правила. То есть, $R_{jk} \in R^*_j \Leftrightarrow \forall C_i \in C'' \exists l : X^{OUT}_{il} \cap X^{OUT}_{jk} = \varnothing \,\&\, X^{OUT}_{il} \cap X^{IN}_{jk} = \varnothing$

6. Инициализируем переменную цикла $k = 0$ для просмотра всех перспективных правил из множества R^*_j.

7. Устанавливаем индекс текущего правила $k = k + 1$. Если $k > \left|R^*_j\right|$ и множество перспективных правил исчерпано ($R^*_j = \varnothing$) при попытке разрешить текущее ограничение, то переходим к пункту 3 с целью анализа другого ограничения.

8. Выбираем правило $R^*_{jk} \in R^*_j$ и добавляем его в план решения $R^*_{jk} \in S$. Перемещаем разрешенное ограничение C''_j из множества C'' в C'. Переходим к следующей итерации алгоритма $i = i + 1$.

9. Если все ограничения разрешены и $C'' = \varnothing$, то **план решения найден**. В противном случае переходим к пункту 2 для дальнейшего планирования решения.

Теорема. Алгоритм распространения значений заканчивает свою работу, либо когда план решения найден, либо когда он не существует.

Доказательство.

□ Алгоритм строит корректный план решения, выстраивая правила разрешения ограничений в порядке, исключающем конфликты.

Если план решения алгоритмом не обнаружен, то его действительно не существует, поскольку в худшем случае алгоритм перебирает все варианты его построения за исключением заведомо некорректных, приводящих к образованию циклов в графе плана решения. ■

Как было отмечено, задача разрешения ограничений в общей постановке является NP-полной. Приведенная алгоритмическая версия метода распространения значений также имеет экспоненциальную оценку сложности

в худшем случае, однако позволяет уменьшить число анализируемых вариантов планирования решения за счет исключения заведомо неперспективных.

3.3. Комбинированный алгоритм разрешения ограничений

Построим комбинированный алгоритм, который гарантировал бы решение задачи в самой общей постановке с использованием метода распространения значений, однако осуществлял бы это более эффективным образом при наличии в задаче особенностей, допускающих применение метода распространения степеней свободы. С этой целью в предлагаемом алгоритме будем привлекать метод распространения степеней свободы всякий раз, когда есть возможность исключить свободные переменные, а при ее отсутствии — использовать универсальный метод распространения значений со спекулятивной оценкой перспективности выбранных правил.

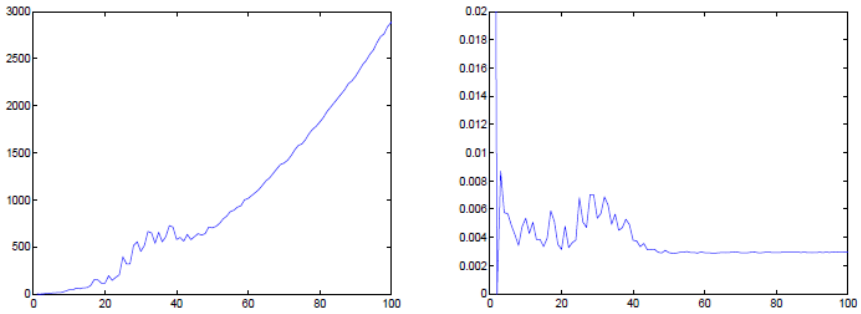
Комбинированный алгоритм работает следующим образом:

1. Определяются переменные, входящие только в одно ограничение исходной системы.
2. Проверяется, есть ли у этих ограничений правиламетоды, которые на выходе имеют только переменные, найденные в п. 1.
3. Если есть, то переменные считаются свободными, и ограничение разрешается по методу, найденному в п. 2.
4. Если свободных переменных нет, то включается поиск решения, основанный на определении неперспективности применения правил. В каждой итерации поиска, если появляется свободная переменная, то соответствующее ограничение разрешается по правилу, выход которого состоит из свободных переменных.

4. Вычислительные эксперименты

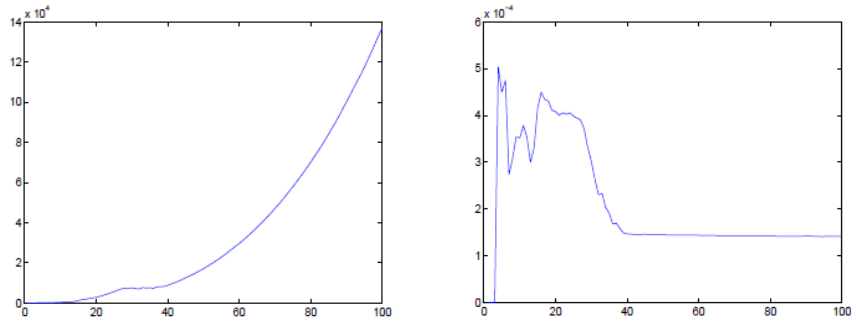
Комбинированный алгоритм, как и базовый метод распространения значений, в худшем случае перебирает все варианты решения за экспоненциальное время. Однако на практике алгоритм демонстрирует полиномиальные оценки сложности в случаях, допускающих эффективную редукцию исходной системы ограничений на основе метода распространения степеней свободы. Ниже приводятся результаты вычислительных экспериментов, демонстрирующие время работы алгоритма в зависимости от числа ограничений и переменных в исходной системе, а также его асимптотическую сложность на наборе тестовых задач.

В качестве первой тестовой задачи была выбрана система ограничений переменной размерности n следующего вида:



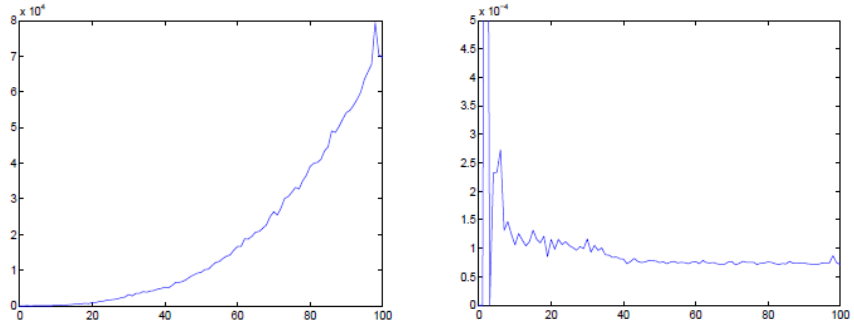
(а) (б)

Рис. 2. Время работы алгоритма в зависимости от количества ограничений (а) и его оценка сложности $O(x^2)$ (б) при прямом порядке задания ограничений



(а) (б)

Рис. 3. Время работы алгоритма в зависимости от количества ограничений (а) и его оценка сложности $O(x^3)$ (б) при обратном порядке задания ограничений



(а) (б)

Рис. 4. Время работы алгоритма в зависимости от количества ограничений (а) и его оценка сложности $O(x^3)$ (б) при произвольном порядке задания ограничений

$$\begin{cases} x_1 = f_1(x_0) \\ x_2 = f_2(x_0, x_1) \\ \dots \\ x_n = f_n(x_0, x_1, \dots, x_{n-1}) \end{cases} \quad (1)$$

Структура зависимостей между переменными данной задачи подобна портрету нижней треугольной матрицы Якоби для соответствующей системы алгебраических уравнений.

Поскольку время работы алгоритма зависит от порядка, в котором задаются ограничения системы и правила их разрешения, было проведено несколько серий экспериментов. Результаты экспериментов с прямым, обратным и произвольным порядком задания ограничений и соответствующих правил разрешения представлены на Рис. 2, 3 и 4 соответственно. Прямой порядок соответствует способу нумерации ограничений в приведенной выше системе сверху вниз (1), обратный — нумерации в противоположном направлении снизу вверх, а произвольный — нумерации на основе матрицы перестановок, сгенерированных случайным образом. Полученные результаты подтверждают полиномиальную сложность комбинированного алгоритма на рассмотренном классе задач, причем алгоритм демонстрирует квадратичную асимптотическую сложность при прямом порядке задания ограничений и характерную кубическую — при обратном и произвольном порядке.

Эксперименты проводились на персональном компьютере типовой конфигурации с процессором Core 2 Duo E8400 (3.00 GHz) и оперативной памятью 2GB (800 MHz). Время работы алгоритма представлено на графиках в миллисекундах CPU.

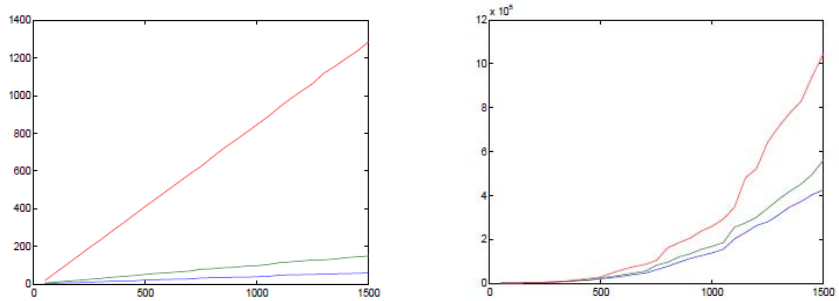
В качестве альтернативной тестовой задачи использовалась система ограничений переменной размерности n с фиксированным максимальным числом параметров k:

$$\begin{cases} x_1 = f_1(x_0) \\ x_2 = f_2(x_0, x_1) \\ \dots \\ x_k = f_k(x_0, x_1, \dots, x_{k-1}) \\ x_{k+1} = f_{k+1}(x_1, x_2, \dots, x_k) \\ \dots \\ x_n = f_n(x_{n-k}, x_{n-k+1}, \dots, x_{n-1}) \end{cases} \quad (2)$$

Выбранная задача эквивалентна системе алгебраических уравнений с диагональной матрицей Якоби шириной k.

Проведенные эксперименты показывают существенное влияние числа параметров в ограничениях на время работы алгоритмов, что объясняется анализом большего числа зависимостей по данным при выборе каждого очередного правила. Не менее важной представляется установленная линейная сложность комбинированного алгоритма для данной задачи в отличие от метода распространения значений. Существенный выигрыш комбинированного алгоритма объясняется эффективной редукцией исходной системы ограничений в результате исключения свободных переменных базовым методом распространения степеней свободы. Как видно из графиков, приведенных на Рис. 5, выигрыш на задачах типовой размерности может составлять несколько порядков и продолжает расти с увеличением размерности задачи.

Наконец, приведем тестовую задачу, которая служит иллюстрацией того, что использование комбинированного алгоритма не налагает каких-либо дополнительных условий на постановку задачи в ограничениях в отличие от алгоритма распространения степеней свободы. Тем самым, более высокая эффективность алгоритма не является следствием сужения границ его применимости или снижения надежности.



(а) (б)

Рис. 5: Время работы комбинированного алгоритма (а) и алгоритма распространения значений (б) в зависимости от количества ограничений при разных значениях параметра k=50, k=10, k=1 (верхние, средние и нижние кривые на графиках соответственно)

Рассмотрим систему ограничений

$$\begin{cases} C_1 : (f_{11}(x_0) - x_1)(f_{12}(x_1) - x_2) = 0 \\ C_2 : (f_{21}(x_1) - x_2)(f_{22}(x_2) - x_3) = 0 \\ \dots \\ C_n : (f_{n1}(x_{n-1}) - x_n) = 0 \end{cases} \quad (3)$$

со следующим набором правил разрешения:

$$\begin{cases} R_{11} : x_1 = f_{11}(x_0);\ R_{12} : x_2 = f_{12}(x_1); \\ R_{21} : x_2 = f_{21}(x_1);\ R_{22} : x_3 = f_{22}(x_2); \\ \dots \\ R_{n1} : x_n = f_{n1}(x_{n-1}); \end{cases} \quad (4)$$

В подобной постановке не выполняется условие вхождения всех переменных ограничения в каждое из его правил, которое необходимо для гарантированного поиска решения с помощью метода распространения степеней свободы. При работе данный алгоритм диагностирует невозможность найти план решения, хотя он существует. Комбинированный алгоритм справляется с задачей благодаря привлечению метода распространения значений. Примечательно, что решение находится за линейное время.

5. Заключение

Таким образом, рассмотрены вопросы применения методов локального распространения к задачам программирования в ограничениях. Предложенный комбинированный алгоритм сочетает в себе элементы метода распространения степеней свободы и метода распространения значений, обеспечивая надежное решение широких классов задач за полиномиальное время. Проведенные серии вычислительных экспериментов и полученные экспериментальные оценки сложности комбинированного алгоритма подтверждают его конкурентные преимущества над традиционными методами, что обуславливает его использование в перспективных системах программирования в ограничениях.

Литература

[1] Dechter R. Constraint processing. Morgan Kaufmann, San Francisco, 2003.
[2] E. Tsang. Foundations of constraint satisfaction. Academic Press Limited, London & San-Diego, 1993.
[3] K. Bharat. Supporting the Construction of Distributed, Interoperative, User Interface Applications. Ph.D. Dissertation, Georgia Institute of Technology, 1996.
[4] R.D. Hill. The Abstraction-Link-View Paradigm: Using Constraints to Connect User Interfaces to Applications. Proceedings of SIGCHI '92. 1992, pp. 335-342.
[5] Joao Pascoal Faria, Raul Moreira Vidal. 1999. Data-driven Active Rules for the Maintenance of Derived Data and Integrity Constraints in User Interfaces to Databases
[6] C. M. Hoffman, A. Lomonosov. Decomposition plans for geometric constraint systems. // J. Symbolic Computation. Volume 31. 2001. pp. 367-408.
[7] W. Bouma, I. Fudos, C. Hoffmann, J. Cai, R. Paige. A Geometric Constraint Solver. 1995.
[8] A. Borning, K. Marriott, P. Stuckey, Y. Xiao. Solving Linear Arithmetic Constraints for User Interface Applications: Algorithm Details. Technical Report 97-06-01, Department of Computer Science and Engineering University of Washington, 1997
[9] M. Chetty, K.P. Dabke. Symbolic computations: an overview and application to controller design. // Proceedings of IEEE sponsored international conference on Information, Decision and Control, Adelaide, 8th-10th February, 1999, pp.451–456.
[10] I. Fudos. Editable Representations For 2D Geometric Design. 1993
[11] B. V. Zanden, An incremental algorithm for satisfying hierarchies of multiway dataflow constraints. // ACM Transactions on Programming Languages and Systems. Volume 18, Issue 1. 1996. 30-72.
[12] A. Borning, R. Anderson, B. Freeman-Benson. Indigo: A Local Propagation Algorithm for Inequality Constraints. // In Proceedings of the 1996 ACM Symposium on User Interface Software and Technology, 1996, pp. 129-136.
[13] M. Sannella. Skyblue: a multi-way local propagation constraint solver for user interface construction. // Proceedings of the 7th annual ACM symposium on User interface software and technology, 1994, pp. 137-146.
[14] J. H. Maloney. Using Constraints for User Interface Construction. Doctoral Thesis. University of Washington. 1992
[15] D. Frigioni, A. Marchetti-Spaccamela, U. Nanni. Dynamic algorithms for classes of constraint satisfaction problems. Theor. Comput. Sci. 259, 1-2 (May. 2001), 2001. pp. 287-305.