УДОВЛЕТВОРЕНИЕ ОГРАНИЧЕНИЙ В ЗАДАЧАХ МАТЕМАТИЧЕСКОГО ПРОГРАММИРОВАНИЯ

В. В. Телерман
Институт систем информатики СО РАН, Новосибирск, Россия
e-mail: telerman@iis.nsk.su

Д. М. Ушаков
РосНИИ искусственного интеллекта, Новосибирск, Россия
e-mail: ushakov@iis.nsk.su

A number of problems to be solved can be formalted as Constraint Satisfaction Problems (CSP). Subdefinite models are the powerful apparatus for solving such problems. The solving means the searching outer estimation of the set of all solutions of CSP. In the paper a modification of the apparatus is considered, which let one to solve the problem of finding the optimal solution of CSP, i. e. mathematical programming problem.

1. Введение

Алгоритмы удовлетворения ограничений относятся к универсальным методам решения задач. Впервые предложенные в начале 1970-х годов в [1] для решения комбинаторных задач на конечных областях, они нашли самое широкое применение. Идея использовать те же самые методы для работы с бесконечными и непрерывными областями привела к понятию интервальных ограничений [2, 3]. Во многом опережая эти работы, в начале 1980-х годов А. С. Нариньяни разработал концепцию недоопределенности и аппарат недоопределенных моделей (н-моделей) [4], развитый затем в других работах [5, 6]. Большинство алгоритмов удовлетворения ограничений служат для вычисления некоторой внешней оценки множества всех решений задачи. В случае интервальных ограничений речь идет об интервальной оценке, то есть о решении задачи, столь популярной в интервальном анализе [7]. В настоящей статье описывается модификация метода удовлетворения ограничений, основанного на аппарате н-моделей, которая позволяет решать задачи математического программирования, то есть находить некоторую интервальную оценку оптимального решения задачи.

2. Удовлетворение ограничений

Определение 1. Пусть X — некоторое произвольное множество, которое мы будем называть *универсальным множеством или универсумом. Недоопределенным расшире-*

нием (н-расширением) универсума X называется любая конечная система ${}^*X$ его подмножеств, содержащая $\emptyset$, X и замкнутая относительно пересечения множеств. Элементы множества ${}^*X$ будем называть *недоопределенными значениями,* а элементы ${}^*X^n$ — *векторами недоопределенных значений.* Каждый вектор $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \in {}^*X^n$ будем рассматривать одновременно как подмножество $\mathbf{x} = \mathbf{x}_1 \times \ldots \times \mathbf{x}_n$ множества $X^n = X \times \ldots \times X$, распространяя на такие векторы обычные теоретико-множественные операции и отношения ($\cap, \cup, \subseteq, \in$ и т. п.) Для произвольного подмножества $S \subseteq X^n$ обозначим ${}^*X(S)$ его *недоопределенное замыкание (недоопределенную оболочку),* наименьший элемент ${}^*X^n$, содержащий S, т. е.

$$ {}^*X(S) = \bigcap_{S \subseteq \mathbf{x} \in {}^*X^n} \mathbf{x}. $$

Пример 1. Пусть $\mathcal{R}$ — множество всех вещественных чисел, а R_0 — конечное подмножество в $\mathcal{R}$. Обозначим $\mathcal{I}(R_0)$ множество всех замкнутых вещественных интервалов (вместе с пустым множеством) с границами из $R_0 \cup \{-\infty, +\infty\}$. Элемент $\mathbf{x} \in \mathcal{I}(R_0)$ представляет собой интервал

$$ \mathbf{x} = [\underline{\mathbf{x}}, \overline{\mathbf{x}}] = \{x \in \mathcal{R} \mid \underline{\mathbf{x}} \le x \le \overline{\mathbf{x}}\}. $$

(Мы полагаем $-\infty < x < +\infty$ для любого $x \in \mathcal{R}$.) При $\underline{\mathbf{x}} > \overline{\mathbf{x}}$ интервал $[\underline{\mathbf{x}}, \overline{\mathbf{x}}]$ представляет собой пустое множество. Понятно, что $[-\infty, +\infty] = \mathcal{R}$ и система $\mathcal{I}(R_0)$ замкнута относительно пересечения интервалов:

$$ [\underline{\mathbf{x}}, \overline{\mathbf{x}}] \cap [\underline{\mathbf{y}}, \overline{\mathbf{y}}] = [\max(\underline{\mathbf{x}}, \underline{\mathbf{y}}), \min(\overline{\mathbf{x}}, \overline{\mathbf{y}})]. $$

Таким образом, $\mathcal{I}(R_0)$ является одним из возможных недоопределенных расширений множества $\mathcal{R}$. Будем называть $\mathcal{I}(R_0)$ *интервальным расширением* множества $\mathcal{R}$. Для произвольного вещественного x обозначим

$$ x^+ = \min\{y \in R_0 \cup \{-\infty, +\infty\} \mid x \le y\}, $$
$$ x^- = \max\{y \in R_0 \cup \{-\infty, +\infty\} \mid x \ge y\}. $$

Тогда *интервальная оболочка* $\mathcal{I}(R_0)(S)$ множества $S \subseteq \mathcal{R}^n$ представляет собой интервальный вектор $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathcal{I}(R_0)^n$, где

$$ \mathbf{x}_i = [(\inf(\pi_i(S)))^-, (\sup(\pi_i(S)))^+], \quad i = 1, \ldots, n. $$

Здесь π_i — оператор проектирования $\mathcal{R}^n$ на i-е одномерное подпространство. Понятно, что интервальная оболочка является недоопределенной оболочкой для недоопределенного расширения $\mathcal{I}(R_0)$.

Определение 2. Рассмотрим произвольное n-арное отношение P над универсумом X, т. е. $P \subseteq X^n$. Пусть ${}^*X$ — произвольное н-расширение X. Сопоставим отношению P *функцию интерпретации* ${}^*P : {}^*X^n \to {}^*X^n$, действующую следующим образом:

$$ {}^*P(\mathbf{x}) = {}^*X(P \cap \mathbf{x}). $$

Смысл функции интерпретации отношения таков. Если n переменных связаны отношением P и каждой переменной сопоставлено ее множество допустимых значений (мы предпочитаем рассматривать это множество как одно недоопределенное значение), то функция интерпретации отношения P “фильтрует” эти множества, исключая из них те значения, принимая которые переменные не будут удовлетворять отношению P.

Определение 3. Пусть X — произвольный универсум, а R — конечное множество отношений различной арности над X. Пару $({}^*X, {}^*R)$, где ${}^*X$ — недоопределенное расширение универсума X, а ${}^*R$ — набор функций интерпретации в ${}^*X$ отношений из R, назовем *недоопределенной моделью* системы (X, R), если каждая функция из ${}^*R$ *эффективно вычислима.*

Рассмотрим отношения над множеством $\mathcal{R}$:

$$\begin{aligned}
\mathrm{val}_a^b &= \{x \in \mathcal{R} \mid a \le x \le b\}, \text{ для } a, b \in \mathcal{R} \cup \{-\infty, +\infty\},\\
\mathrm{eq} &= \{(x, y) \in \mathcal{R}^2 \mid x = y\},\\
\mathrm{le} &= \{(x, y) \in \mathcal{R}^2 \mid x \le y\},\\
\mathrm{abs} &= \{(x, y) \in \mathcal{R}^2 \mid |x| = y\},\\
\exp &= \{(x, y) \in \mathcal{R}^2 \mid e^x = y\},\\
\sin &= \{(x, y) \in \mathcal{R}^2 \mid \sin x = y\},\\
\mathrm{add} &= \{(x, y, z) \in \mathcal{R}^3 \mid x + y = z\},\\
\mathrm{mul} &= \{(x, y, z) \in \mathcal{R}^3 \mid x * y = z\}.
\end{aligned}$$

Нетрудно убедиться в справедливости следующего утверждения.

Предложение 1. *В интервальном расширении множества $\mathcal{R}$ (пример 1) существуют эффективные алгоритмы вычисления функций* ${}^*\mathrm{val}_a^b$, ${}^*\mathrm{eq}$, ${}^*\mathrm{le}$, ${}^*\mathrm{abs}$, ${}^*\exp$, ${}^*\sin$, ${}^*\mathrm{add}$, ${}^*\mathrm{mul}$.

Некоторые из них можно записать в виде формул:

$$\begin{aligned}
{}^*\mathrm{val}_a^b(\mathbf{x}) &= [\max(a, \underline{\mathbf{x}}), \min(b, \overline{\mathbf{x}})],\\
{}^*\mathrm{eq}(\mathbf{x}, \mathbf{y}) &= ([\max(\underline{\mathbf{x}}, \underline{\mathbf{y}}), \min(\overline{\mathbf{x}}, \overline{\mathbf{y}})], [\max(\underline{\mathbf{x}}, \underline{\mathbf{y}}), \min(\overline{\mathbf{x}}, \overline{\mathbf{y}})]),\\
{}^*\mathrm{le}(\mathbf{x}, \mathbf{y}) &= ([\underline{\mathbf{x}}, \min(\overline{\mathbf{x}}, \overline{\mathbf{y}})], [\max(\underline{\mathbf{x}}, \underline{\mathbf{y}}), \overline{\mathbf{y}}]),\\
{}^*\mathrm{add}(\mathbf{x}, \mathbf{y}, \mathbf{z}) &= ([\max(\underline{\mathbf{x}}, (\underline{\mathbf{z}} - \overline{\mathbf{y}})^-), \min(\overline{\mathbf{x}}, (\overline{\mathbf{z}} - \underline{\mathbf{y}})^+)],\\
&\quad\ [\max(\underline{\mathbf{y}}, (\underline{\mathbf{z}} - \overline{\mathbf{x}})^-), \min(\overline{\mathbf{y}}, (\overline{\mathbf{z}} - \underline{\mathbf{x}})^+)],\\
&\quad\ [\max(\underline{\mathbf{z}}, (\underline{\mathbf{x}} + \underline{\mathbf{y}})^-), \min(\overline{\mathbf{z}}, (\overline{\mathbf{x}} + \overline{\mathbf{y}})^+)]).
\end{aligned}$$

Очевидно, что произвольную систему алгебраических уравнений и неравенств над вещественными числами можно представить в виде некоторого набора введенных выше примитивных отношений (возможно, для этого придется ввести в систему дополнительные переменные). При этом с каждым m-арным отношением связываются m переменных, являющихся его аргументами. Назовем такую конструкцию *ограничением*, а набор ограничений — *задачей удовлетворения ограничений*. Ниже мы приводим формальное определение этих понятий.

Определение 4. Пусть X — некоторый универсум. *Задача удовлетворения ограничений* (ЗУО) над X с n переменными задается конечным набором ограничений $C = \cup_{m=1}^{M} C_m$. Каждое ограничение $c \in C_m$ представляет собой пару $c = (r_c, \pi_c)$, где $r_c \subseteq X^m$ — произвольное m-арное отношение на X, а $\pi_c : X^n \to X^m$ — функция, выполняющая проектирование вектора $x = (x_1, \ldots, x_n) \in X^n$ на некоторые его m компонент, т. е. $\pi_c(x) = (x_{i_1}, \ldots, x_{i_m})$ для некоторых $i_1, \ldots, i_m \in \{1, \ldots, n\}$. Обозначим $\arg c$ множество индексов $\{i_1, \ldots, i_m\}$, на которые функция π_c выполняет проектирование. Если переменные задачи обозначить именами $x_1, \ldots, x_n$, то ограничение $c = (r_c, \pi_c) \in C_m$ можно короче записать в виде

$$r_c(x_{i_1}, \ldots, x_{i_m}).$$

Множество решений $\Sigma(C)$ задачи удовлетворения ограничений C над X с n переменными определяется следующим образом:

$$\Sigma(C) = \{x \in X^n \mid (\forall c \in C) \quad \pi_c(x) \in r_c\}.$$

Пример 2. Систему уравнений над вещественными числами

$$\begin{cases} x + y = 6, \\ 2x = y \end{cases}$$

можно описать в виде задачи удовлетворения ограничений с четырьмя переменными x_1, x_2, x_3, x_4:

$$C = \{\mathrm{add}(x_1, x_2, x_3), \mathrm{val}_6^6(x_3), \mathrm{mul}(x_4, x_1, x_2), \mathrm{val}_2^2(x_4)\}.$$

Ясно, что в общем случае не существует алгоритма нахождения множества $\Sigma(C)$ или хотя бы произвольного его элемента. Тем не менее, если построена н-модель системы (X, R), где R включает в себя все отношения, встречающиеся в C (для вещественных чисел такую н-модель легко построить, используя интервальное расширение), то можно предложить универсальный эффективный алгоритм нахождения внешней оценки множества $\Sigma(C)$. Алгоритм является естественным обобщением алгоритма установления локальной совместности сети ограничений AC-2, предложенного для решения комбинаторных задач на конечных областях в [1].

В целях упрощения дальнейшего изложения расширим функции ${}^*r_c : {}^*X^m \to {}^*X^m$ на пространство ${}^*X^n$. Для этого введем функции ${}^*r_c^+ : {}^*X^n \to {}^*X^n$. Пусть $\pi_c(x_1, \ldots, x_n) = (x_{i_1}, \ldots, x_{i_m})$, а ${}^*r_c(\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_m}) = (\mathbf{y}_1, \ldots, \mathbf{y}_m)$. Тогда

$${}^*r_c^+(\mathbf{x}_1, \ldots, \mathbf{x}_n) = (\mathbf{z}_1, \ldots, \mathbf{z}_n),$$

где

$$\mathbf{z}_i = \begin{cases} \bigcap_{j:i_j=i} \mathbf{y}_j, & \text{если } i \in \arg c, \\ \mathbf{x}_i, & \text{иначе.} \end{cases}$$

Алгоритм 1. Обозначим на шаге t

$$\mathbf{x}^{(t)} \in {}^*X^n - \text{вектор недоопределенных значений},$$
$$Q^{(t)} \subseteq C - \text{множество активных ограничений}.$$

Шаг 0.

$$\mathbf{x}^{(0)} := (X, \ldots, X),$$
$$Q^{(0)} := C.$$

Шаг $t + 1$. Если $Q^{(t)} = \emptyset$, то СТОП.
Иначе выбрать произвольно $c \in Q^{(t)}$,

$$\mathbf{x}^{(t+1)} := {}^*r_c^+(\mathbf{x}^{(t)}),$$
$$Q^{(t+1)} := Q^{(t)} \cup \{d \in C \mid (\exists i \in \arg d) \quad \mathbf{x}_i^{(t+1)} \neq \mathbf{x}_i^{(t)}\} \setminus \{c\}.$$

В работе [5] мы показали справедливость следующих утверждений об алгоритме 1, которые приводим здесь без доказательства из-за ограничений объема.

Предложение 2. *1. Алгоритм 1 всегда завершается, число его итераций можно оценить сверху величиной* $|C|nI(^*X)$*, где* $I(^*X)$ *— длина наибольшей цепи строго вложенных друг в друга элементов* $^*X$*.*

2. Если $\mathbf{x}^*$ *— результат работы алгоритма (т. е. значение* $\mathbf{x}^{(t)}$ *на последнем шаге* t*), то* $\mathbf{x}^* \supseteq \Sigma(C)$*.*

К сожалению, в общем случае мы ничего не можем сказать о том, насколько близко $\mathbf{x}^*$ к $\Sigma(C)$, более того, непустота $\mathbf{x}^*$ не гарантирует непустоту $\Sigma(C)$. Были исследованы различные виды ЗУО — системы линейных уравнений [8], алгебраические уравнения n-й степени [9], но даже в таких частных случаях пока еще далеко не все ясно. Достоинством алгоритма является его универсальность: он применим к произвольным ЗУО, независимо от их сложности. Кроме того, алгоритм является эффективным: требуемое им время вычислений полиномиально зависит от числа переменных и ограничений. Еще одно достоинство алгоритма 1 заключается в том, что на любом шаге t его работы выполняется соотношение $\mathbf{x}^{(t)} \supseteq \Sigma(C)$, т. е. алгоритм 1 является последовательно гарантирующим согласно классификации, предложенной в работе [7]. Зачастую благодаря полученной алгоритмом 1 оценке $\mathbf{x}^*$ множества $\Sigma(C)$ к задаче становятся применимы различные специализированные методы решения, которые были неприменимы (или неэффективны) в исходной постановке. Мы, однако, рассмотрим еще один универсальный метод, который, будучи примененным совместно с описанным выше алгоритмом, позволяет в ряде случаев существенно улучшить оценку $\mathbf{x}^*$.

3. Метод бисекции

Часто область $\mathbf{x}^*$ удается существенно уменьшить с помощью *метода бисекции.*

Определение 5. Назовем *бисекцией* вектора недоопределенных значений $\mathbf{x} \in {}^*X^n$ по i-й компоненте $(1 \le i \le n)$ его разбиение на две части $\mathbf{x}^1, \mathbf{x}^2 \in {}^*X^n$ таким образом, что

$$\begin{gathered}\mathbf{x}_i^1, \mathbf{x}_i^2 \neq \emptyset,\\ \mathbf{x}_i^1, \cap \mathbf{x}_i^2 \neq \mathbf{x}_i,\\ \mathbf{x}_i^1 \cup \mathbf{x}_i^2 = \mathbf{x}_i,\\ \mathbf{x}_j^1 = \mathbf{x}_j^2 = \mathbf{x}_j \text{ для } j \neq i.\end{gathered}$$

Разделим множество $^*X^n$ на три части: *пустые векторы,* которые содержат пустое множество как компоненту*, разделяемые векторы,* для которых возможно проведение бисекции по какой-нибудь компоненте, и *элементарные векторы,* для которых такая бисекция невозможна.

Алгоритм 2. Обозначим на каждом шаге t

$$\begin{aligned}&\mathbf{x}^{(t)} \in {}^*X^n - \text{вектор недоопределенных значений},\\ &S^{(t)} \subseteq {}^*X^n - \text{множество отложенных векторов},\\ &R^{(t)} \subseteq {}^*X^n - \text{множество найденных “решений”}.\end{aligned}$$

Шаг 0.

$$\begin{aligned}\mathbf{x}^{(0)} &:= (X, \ldots, X),\\ S^{(0)} &:= \emptyset,\\ R^{(0)} &:= \emptyset.\end{aligned}$$

Шаг $t+1$. Применить алгоритм 1 к $\mathbf{x}^{(t)}$, получив на выходе значение $\mathbf{x}^{(t)*}$.
Если $\mathbf{x}^{(t)*}$ — разделяемый вектор, то провести бисекцию вектора $\mathbf{x}^{(t)*}$ на векторы $\mathbf{x}^1, \mathbf{x}^2$,

$$\begin{aligned}\mathbf{x}^{(t+1)} &:= \mathbf{x}^1,\\ S^{(t+1)} &:= S^{(t)} \cup \{\mathbf{x}^2\},\\ R^{(t+1)} &:= R^{(t)}.\end{aligned}$$

Иначе выбрать $\mathbf{x} \in S^{(t)}$ (если $S^{(t)} = \emptyset$, то СТОП),

$$\begin{aligned}\mathbf{x}^{(t+1)} &:= \mathbf{x},\\ S^{(t+1)} &:= S^{(t)} \setminus \{\mathbf{x}\},\\ R^{(t+1)} &:= \begin{cases} R^{(t)}, \text{ если } \mathbf{x}^{(t)*} \text{— пустой},\\ R^{(t)} \cup \{\mathbf{x}^{(t)^*}\}, \text{ иначе.}\end{cases}\end{aligned}$$

Результатом алгоритма после его завершения на N-м шаге следует считать множество $\mathrm{Sol}^{(N)}$. Каждый элемент данного множества — вектор элементарных недоопределенных значений, которые невозможно разделить на две части в рассматриваемом н-расширении. В случае интервального расширения множества $\mathcal{R}$ речь идет о максимально "узких" n-мерных параллелепипедах пространства $\mathcal{R}^n$. Величину "узости" можно регулировать, выбирая различные множества R_0, из которых формируются границы интервалов: чем больше элементов в R_0 (чем выше точность представления интервалов), тем более "узкие" параллелепипеды будут получаться на выходе алгоритма 2. Алгоритм 2 не гарантирует, что каждый из этих параллелепипедов содержит решение ЗУО, но в силу их "узости" существование решения можно проверить другими методами, характерными для задач данного класса.

Сформулируем без доказательства следующее утверждение об алгоритме 2.

Предложение 3. *1. Алгоритм 2 всегда завершается, но количество шагов в общем случае экспоненциально зависит от количества переменных ЗУО.*

2. Если $R^{(N)}$ — результат работы алгоритма 2, то

$$\bigcup_{\mathbf{x} \in R^{(N)}} \mathbf{x} \supseteq \Sigma(C).$$

Далее рассмотрим применения описанного аппарата к задачам математического программирования.

4. Поиск оптимального решения

Определение 6. Пусть X — универсум с линейным порядком $\leq$, C — ЗУО над X с n переменными. Решение $\hat{x} = (\hat{x}_1, \ldots, \hat{x}_n) \in \Sigma(C)$ назовем *оптимальным,* если $\hat{x}_1 \leq x_1$ для любого $x = (x_1, \ldots, x_n) \in \Sigma(C)$.

Любую задачу математического программирования можно свести к задаче поиска оптимального решения ЗУО над соответствующими универсумами. При поиске оптимального решения ЗУО удобнее выделять не целевую функцию, а целевую переменную (x_1).

Первый рассматриваемый нами метод для поиска оптимального решения основан на понятии *упорядоченной бисекции*.

Определение 7. Бисекция вектора недоопределенных значений $\mathbf{x} \in {}^*X^n$ по i-й компоненте на векторы $\mathbf{x}^1 \in {}^*X^n$ и $\mathbf{x}^2 \in {}^*X^n$ называется *упорядоченной,* если $x^1 \leq x^2$ для любых $x^1 \in \mathbf{x}^1_i$, $x^2 \in \mathbf{x}^2_i$.

Идея алгоритма поиска оптимального решения состоит в применении последовательности упорядоченных бисекций вектора недоопределенных значений по первой компоненте и организации множества отложенных векторов $S^{(t)}$ в виде стека, позволяющего выбирать отложенные состояния не произвольно, а в порядке, обратном их "откладыванию".

Алгоритм 3. Обозначим на каждом шаге t

$$\mathbf{x}^{(t)} \in {}^*X^n - \text{вектор недоопределенных значений},$$
$$S^{(t)} \subseteq {}^*X^n - \text{множество отложенных векторов}.$$

Шаг 0.

$$\mathbf{x}^{(0)} := (X, \ldots, X),$$
$$S^{(0)} := \emptyset.$$

Шаг $t+1$. Применить алгоритм 1 к $\mathbf{x}^{(t)}$, получив на выходе значение $\mathbf{x}^{(t)*}$.

Если $\mathbf{x}^{(t)*}$ — разделяемый вектор, то провести упорядоченную бисекцию вектора $\mathbf{x}^{(t)*}$ по первой компоненте на векторы $\mathbf{x}^1$, $\mathbf{x}^2$ (если такая бисекция невозможна, то провести бисекцию по любой другой компоненте),

$$\mathbf{x}^{(t+1)} := \mathbf{x}^1,$$
$$S^{(t+1)} := S^{(t)} \cup \{\mathbf{x}^2\}.$$

Иначе если $\mathbf{x}^{(t)*}$ — непустой вектор, то СТОП "РЕШЕНИЕ $\mathbf{x}^{(t)*}$", иначе выбрать последний отложенный $\mathbf{x} \in S^{(t)}$ (если $S^{(t)} = \emptyset$, то СТОП "НЕТ РЕШЕНИЙ"),

$$\mathbf{x}^{(t+1)} := \mathbf{x},$$
$$S^{(t+1)} := S^{(t)} \setminus \{\mathbf{x}\}.$$

Предложение 4. *1. Алгоритм 3 всегда завершается, количество его шагов в общем случае экспоненциально зависит от* n.

2. Если ЗУО C *имеет оптимальное решение* $\hat{x}$*, то на любом шаге* t *алгоритма 3 имеет место соотношение*

$$\inf_{x \in \mathbf{x}^{(t)}} x_1 \leq \hat{x}_1.$$

Конечно, получаемый в результате работы алгоритма 3 элементарный вектор недоопределенных значений $\mathbf{x}^*$ не обязательно содержит решение ЗУО C. Проверку существования

решения внутри $\mathbf{x}^*$ следует проводить другими методами. Если решение внутри $\mathbf{x}^*$ не обнаружено, то необходимо продолжить работу алгоритма 3 (выбрав последний отложенный вектор в множестве $S^{(t)}$).

Заметим, что скорость поиска "точного" решения (не обязательно оптимального) методом бисекции существенно зависит от порядка применения бисекций к компонентам вектора недоопределенных значений. Одна из наиболее полезных эвристик выбора компонент вектора недоопределенных значений для бисекции состоит в выборе компоненты, соответствующей переменной с наиболее узким множеством допустимых значений или входящей в наибольшее число ограничений из множества C. В этом смысле алгоритм 3 имеет существенный недостаток, связанный с необходимостью проведения бисекций по первой компоненте (соответствующей целевой переменной), что во многих задачах может оказаться неоптимальным. Поэтому мы рассмотрим здесь еще один алгоритм (очень похожий на известный метод "ветвей и границ") для поиска оптимального решения ЗУО.

Алгоритм 4. Обозначим на каждом шаге t

$$\begin{aligned}
\mathbf{x}^{(t)} &\in {}^*X^n - \text{вектор недоопределенных значений},\\
S^{(t)} &\subseteq {}^*X^n - \text{множество отложенных векторов},\\
C^{(t)} &- \text{множество ограничений модели},\\
\hat{\mathbf{x}}^{(t)} &\in {}^*X^n - \text{лучшее из найденных решений}.
\end{aligned}$$

Шаг 0.

$$\begin{aligned}
\mathbf{x}^{(0)} &:= (X, \ldots, X),\\
S^{(0)} &:= \emptyset,\\
C^{(0)} &:= C,\\
\hat{\mathbf{x}}^{(0)} &:= \mathbf{x}^{(0)}.
\end{aligned}$$

Шаг $t+1$. Применить алгоритм 1 к $\mathbf{x}^{(t)}$, получив на выходе значение $\mathbf{x}^{(t)*}$.

Если $\mathbf{x}^{(t)*}$ — разделяемый вектор, то провести бисекцию вектора $\mathbf{x}^{(t)*}$ (по любой компоненте, не обязательно упорядоченную) на векторы $\mathbf{x}^1$, $\mathbf{x}^2$,

$$\begin{aligned}
\mathbf{x}^{(t+1)} &:= \mathbf{x}^1,\\
S^{(t+1)} &:= S^{(t)} \cup \{\mathbf{x}^2\},\\
\hat{\mathbf{x}}^{(t+1)} &:= \hat{\mathbf{x}}^{(t)}.
\end{aligned}$$

Иначе выбрать произвольный $\mathbf{x} \in S^{(t)}$ (если $S^{(t)} = \emptyset$, то СТОП),

$$\begin{aligned}
\mathbf{x}^{(t+1)} &:= \mathbf{x},\\
S^{(t+1)} &:= S^{(t)} \setminus \{\mathbf{x}\},\\
C^{(t+1)} &:= \begin{cases} C^{(t)} \cup \{\mathrm{val}_{-\infty}^{M}(x_1)\} \text{ (где } M = \sup \mathbf{x}_1^{(t)*}) \text{ , если } \mathbf{x}^{(t)*} \neq \emptyset, \\ C^{(t)}, \text{ иначе.} \end{cases},\\
\hat{\mathbf{x}}^{(t+1)} &:= \begin{cases} \mathbf{x}^{(t)^*}, \text{ если } \mathbf{x}^{(t)*} \neq \emptyset, \\ \hat{\mathbf{x}}^{(t)}, \text{ иначе.} \end{cases}
\end{aligned}$$

Алгоритм 4, в отличие от алгоритма 3, идет к оптимальному решению не от меньших значений оптимума к большим, а от больших к меньшим. Сформулируем следующее утверждение о его свойствах.

Предложение 5. *1. Алгоритм 4 всегда завершается, количество его шагов в общем случае экспоненциально зависит от* n.

2. Если ЗУО C *имеет оптимальное решение* $\hat{x}$, *то на любом шаге* t *алгоритма 3 имеет место соотношение*

$$\sup_{x \in \hat{\mathbf{x}}^{(t)}} x_1 \geq \hat{x}_1.$$

Алгоритм 4, так же как и алгоритм 3, не является полным, т. е. получаемый в результате его работы элементарный вектор $\hat{\mathbf{x}}^*$ не обязательно содержит оптимальное решение ЗУО C. Если необходима полнота, то на каждом шаге алгоритма 4 необходимо проверять существование решения внутри $\hat{\mathbf{x}}^{(t)*}$.

Таким образом, алгоритмы 3 и 4 можно отнести к последовательно гарантирующим в следующем смысле: на любом шаге вычислений их можно прервать, при этом алгоритм 3 даст нам нижнюю оценку значения целевой переменной, а алгоритм 4 — верхнюю.

5. Заключение

Сотрудниками ИСИ СО РАН им. А.П.Ершова и РосНИИ ИИ совместно разработан программный комплекс НеМо+, предназначенный для решения задач удовлетворения ограничений на основе аппарата н-моделей. Комплекс позволяет решать ЗУО над различными областями: вещественными, целыми, булевыми, строковыми, составными (такими, как множества или массивы), допуская использование различных областей в рамках одной задачи. При этом ЗУО специфицируются на языке математических формул и выражений, понятных любому неспециалисту. Все описанные выше особенности решения задач математического программирования реализованы в рамках данного комплекса. Эксперименты показали, что во многих случаях удается получать хорошие оценки решения различных задач. При этом лучшие результаты достигаются на задачах целочисленного и смешанного программирования с большим количеством нелинейных ограничений. В то же время универсальные алгоритмы, предложенные в настоящей работе, часто проигрывают специализированным алгоритмам (например, симплекс-методу) при решении конкретных классов задач (таких, как задачи линейного программирования).

Авторы благодарят анонимного рецензента настоящей статьи за ценные замечания, позволившие нам улучшить качество представленной работы.

Список литературы

[1] Mackworth A. K. Consistency in Networks of Relations. *Artificial Intelligence*, **8**, No. 1, 1977, 99–118.

[2] Hyvonen E. Constraint reasoning based on interval arithmetic: the tolerance propagation approach. *Ibid.*, **58**, 1992, 71–112.

[3] Benhamou F., Older W. J. Applying Interval Arithmetic to Real, Integer and Boolean Constraints. *J. of Logic Programming*, **32**. No 1, 1997, 1–24.

[4] Нариньяни А. С. *Недоопределенные модели и операции с недоопределенными значениями.* Препринт ВЦ СО АН СССР, №400, Новосибирск, 1982.

[5] Телерман В. В., Ушаков Д. М. Недоопределенные модели: формализация подхода и перспективы развития. В *"Пробл. представления и обработки не полностью определенных знаний"*. Ред. И. Е. Швецов, РосНИИ ИИ, М. — Новосибирск, 1996, 7–30.

[6] Телерман В. В., Сидоров В. А., Ушаков Д. М. Интервальные и мультиинтервальные расширения в недоопределенных моделях. *Вычисл. технологии*, **1**, №2, 1997, 62–70.

[7] Shokin Y. I. On Interval Problems, Interval Algorithms and Their Computational Complexity. In *"Scientific Computing and Validated Numerics"*, Akademie Verlag, Berlin, 1996, 314–328.

[8] Петров Е. С. Сходимость метода недоопределенных вычислений при решении линейных уравнений. В *"Проблемы представления и обработки не полностью определенных знаний"*, РосНИИ ИИ, М. — Новосибирск, 1996, 38–47.

[9] Кашеварова Т. П., Семенов А. Л. Некоторые вопросы сходимости метода недоопределенных вычислений. *Там же*, 31–37.

[10] Shvetsov I., Telerman V., Ushakov D. NeMo+: Object-Oriented Constraint Programming Environment Based on Subdefinite Models. *Lect. Notes in Comp. Sci.*, **1330**, Springer Verlag, Berlin et al., 1997, 534–548.

Поступила в редакцию 27 ноября 1997 г.,
в переработанном виде 12 марта 1998 г.