О задачах оптимального распределения ресурсов и проверки устойчивости для схем функциональных элементов в k-значной логике

А. А. Лебедев

Технология информационного мониторинга была разработана для анализа сложных, слабоформализованных проблем (процессов) на основе всей доступной информации, построения прогнозов их развития и выработки рекомендаций по управлению их развитием. В настоящей работе технология информационного мониторинга формализуется с использованием классического аппарата дискретной математики — схем функциональных элементов и функций k-значной логики. В этой формализации решаются две ключевые задачи технологии информационного мониторинга — проверка устойчивости модели и задача оптимального распределения ресурсов.

Ключевые слова: информационный мониторинг, оптимальное распределение ресурсов, устойчивость дискретных систем.

1. Введение

1.1. Описание технологии информационного мониторинга

Технология информационного мониторинга [7] была разработана для анализа сложных, слабоформализованных проблем (процессов) на основе всей доступной информации, построения прогнозов их развития и выработки рекомендаций по управлению их развитием. Системы, разработанные на базе этой технологии, позволяют иметь развивающуюся во времени модель проблемы на основе оценок

аналитиков, подкрепленную ссылками на все информационные материалы, выбранные ими, с общими и частными оценками состояния проблемы и/или ее аспектов. Использование времени как параметра системы позволяет проводить как ретроспективные анализ, так и строить прогнозы развития проблемы (отвечать на вопросы типа «Что будет, если... ?»). В последнем случае возникает возможность выделения «критических путей» — таких элементов модели, изменение которых может вызвать существенные изменения в состоянии всей проблемы. Знание таких элементов имеет большое практическое значение и позволяет выявить «слабые места» в проблеме на текущий момент времени, разработать мероприятия по блокированию нежелательных ситуаций или провоцированию желательных, то есть в некоторой степени управлять развитием проблемы в интересах организации, ее отслеживающей.

Основой любой системы информационного мониторинга является модель проблемы или процесса. Модель представляет собой ациклический граф, каждой вершине которого, соответствующей некоторому аспекту проблемы, сопоставлена дискретная или лингвистическая [1] переменная, причём значение вершины-«родителя» может как вычисляться на основе значений вершин-«потомков» и заданных операторов агрегирования информации (проблеме выбора адекватных операторов агрегирования посвящена работа [3]), так и назначаться пользователем.

1.2. Формализация и обозначения

Теоретической моделью системы служит Схема Функциональных Элементов (СФЭ) [4]. Функциональные элементы вычисляют функции k-значной логики. Такая формализация полностью описывает случай дискретных систем; в нечётком случае моделируется только набор «если — то» правил, составляющий основу оператора агрегирования, а такие аспекты, как выбор функций принадлежности и метода нечёткого логического вывода, игнорируются.

СФЭ, используемые при моделировании систем информационного мониторинга, удовлетворяют следующим ограничениям:

- схема обладает единственным выходом. Функцию, реализуемую схемой на этом выходе, будем обозначать $F(x_1, \ldots, x_N)$;

- число переменных у функциональных элементов ограничено сверху (максимальное число переменных обозначим n).

Отдельно будет рассматриваться граф СФЭ. Его мы будем обозначать $G = (V, E)$. Являясь графом СФЭ с введёнными выше ограничениями, он обладает следующими свойствами:

- граф ориентирован (выберем ориентацию рёбер от переменных к функциональным элементам);
- граф не содержит ориентированных циклов;
- граф обладает единственной вершиной с нулевой полустепенью исхода. Эту вершину будем называть корневой и обозначать s;
- граф связен (в слабом смысле, то есть без учёта ориентации рёбер; более точно — любая вершина соединена ориентированным путём с корневой);
- полустепень захода вершин ограничена (константой n). Отсюда следует, что $|E| \leqslant n|V|$ (то есть граф является разреженным).

Для вершин графа и их взаимного расположения будем использовать терминологию, используемую для графов-деревьев, но учитывая обратную ориентацию рёбер. Так, вершины с нулевой полустепенью захода будем называть концевыми вершинами или листьями. Если существует ориентированный путь от вершины u к вершине v, v называем родителем u, а u — потомком v, или вершиной, подчинённой v. Если этот путь состоит из одного ребра, то v называем непосредственным родителем u, u — непосредственным потомком v, или вершиной, непосредственно подчинённой v. Существенное отличие от графов-деревьев заключается в том, что у вершины может быть несколько непосредственных родителей.

При практическом построении и использовании систем информационного мониторинга возможно использование неоднородных функций (k — «значность» логики — различна для разных элементов в пределах одной схемы, см. например [2]). В этой работе мы будем считать k постоянным, так как это значительно упрощает обозначения, а переход к неоднородному случаю тривиален. Если это не оговорено отдельно, считаем, что $k \geqslant 3$.

1.3. Структура работы

Работа состоит из трёх разделов. В первом разделе рассматривается задача проверки устойчивости для схем функциональных элементов в k-значной логике. Во втором разделе исследуется задача оптимального распределения ресурсов. Для обеих задач доказывается их труднорешаемость в общем случае, а также приводятся частные случаи, допускающие разрешимость за полиномиальное время. В третьем разделе приводится алгоритм анализа графов схем функциональных элементов, также позволяющий выделить полиномиально разрешимые случаи для обеих задач. В начале каждого раздела перечисляются основные результаты, а их доказательства приводятся далее в разделе.

2. Задача проверки устойчивости

Важным аспектом практического использования любой модели является ее устойчивость. Достаточно очевидно, что при практическом применении любой модели и идентификации ее параметров возникают (маленькие) ошибки измерения. Если модель чувствительна к такого рода естественным ошибкам, вопрос ее практического применения является проблематичным. Технология информационного мониторинга изначально ориентирована на обработку информации разной степени достоверности. Предполагается, что входные данные могут быть неточны. Поэтому необходимо иметь оценку того, как погрешности различной величины во входных данных отразятся на результате, возвращаемом системой.

2.1. Основные результаты

Формальная постановка задачи в выбранной формализации имеет следующий вид: Дана $F(x_1, \ldots, x_N)$ — функция k-значной логики, заданная схемой функциональных элементов над базисом, состоящим из всех функций от n и менее переменных. Для заданного $1 \leqslant A \leqslant k-2$ необходимо проверить, удовлетворяет ли F следующему условию (назовём его A-устойчивостью):

$$\forall \alpha, \beta \in E_k^N \max_{i=1,\ldots,N} |\alpha_i - \beta_i| \leqslant A \Rightarrow |F(\alpha_1, \ldots, \alpha_N) - F(\beta_1, \ldots, \beta_N)| \leqslant A.$$

Класс всех A-устойчивых функций обозначим R_k^A, а класс всех A-устойчивых функций от n (фиксированных) переменных обозначим $R_k^A(n)$. Следующая теорема даёт оценку мощности рассматриваемых классов.

Теорема 1. $\frac{k-A}{A+1}(A+1)^{k^n} \leqslant |R_k^A(n)| \leqslant \frac{k-A}{2A+1}(A+1)^{k^n}(2A+1)^{\lceil \frac{k-1}{A} \rceil^n}$.

Следствие 1.1. $ln|R_k^A(n)| \sim k^n \sim ln|P_k(n)|$.

Следующие две теоремы утверждают, что класс R_k^A является критериальным относительно разрешимости задачи проверки устойчивости эффективным алгоритмом. Из замкнутости классов R_k^A непосредственно следует достаточное условие A-устойчивости функции, реализованной схемой.

Теорема 2. *Если все функции, вычисляемые элементами схемы, A-устойчивы, то реализуемая схемой функция также A-устойчива.*

Теорема 3. $\forall f(x_1, \ldots, x_m), m \leqslant n, f \in P_k \setminus R_k^A$ *задача проверки A-устойчивости для функций, заданных СФЭ над множеством* $f \cup R_k^A(2) \cup R_k^A(1)$, *co-NP-полна.*

Предыдущие результаты показывают, что внесение нетривиальных ограничений на базис СФЭ не приводит к существенному упрощению задачи проверки устойчивости. Следующая теорема показывает, что, внеся ограничения на графовую структуру СФЭ, задачу можно решать за линейное (от числа элементов схемы) время.

Теорема 4. *Для СФЭ, граф которых является деревом, существует алгоритм, решающий задачу проверки устойчивости, за время $C(n)\cdot|V|$, где $|V|$ — число вершин, $C(n)$ — величина, зависящая только от n — максимального числа переменных базисной функции.*

2.2. Мощность классов устойчивых функций

В этом разделе мы докажем теорему 1.

Нижняя оценка: возьмём множество функций, принимающих значения из одного из множеств $\{B, B+1, \ldots, B+A\}$, $B = 0, 1, \ldots, k-A-1$, таких, что $f(0, \ldots, 0) = B$. Все функции принадлежат R_A, число

функций для каждого B — $(A+1)^{k^n-1}$, множества функций, соответствующие разным значениям B, не пересекаются.
Верхняя оценка: разобьём множество E_k^n на параллелепипеды:

$$P_l = \{(x_1, \ldots, x_n) : Al_i \leqslant x_i \leqslant \min(A(l_i+1), k-1)\},$$

где $l = (l_1, \ldots, l_n), l_i = 0, \ldots, \lceil \frac{k-1}{A} \rceil - 1$. Их число — $\lceil \frac{k-1}{A} \rceil^n$. Каждому параллелепипеду сопоставим множество значений $\{B, B+1, \ldots, B+A\}$. Учитывая, что соседние параллелепипеды имеют общие вершины, получаем, что для соответствующих B_1 и B_2 выполняется $|B_1 - B_2| \leqslant A$. Таким образом, число возможных распределений множеств значений не превосходит $(k-A)(2A+1)^{\lceil \frac{k-1}{A} \rceil^n - 1}$ (для параллелепипеда $P_{(0,\ldots,0)}$ множество значений выбирается произвольно, для остальных — с учётом множества значений одного соседнего параллелепипеда).

Для каждого распределения множеств значений число соответствующих ему функций не превосходит $(A+1)^{k^n-1}$, откуда и следует верхняя оценка. Теорема доказана.

2.3. Труднорешаемость задачи проверки устойчивости

Для доказательства теоремы 3 сначала сформулируем и докажем две леммы.

Лемма 1 (свойства классов R_k^A). *1)* $\forall l = 2, \ldots, \lfloor \frac{k-2}{A} \rfloor : R_k^A \subset R_k^{lA}$, *в частности* $R_k^1 \subset R_k^l \forall l = 2, \ldots, k-2$;
2) $\forall B = A+1, \ldots, 2A-1 : R_k^A \not\subseteq R_k^B$.

Доказательство.

1. Рассмотрим произвольную функцию $f \in R_k^A$ (пусть она будет иметь m переменных) произвольную пару наборов $\alpha, \beta \in E_k^m$ таких, что $\max\limits_{i=1,\ldots,m} |\alpha_i - \beta_i| \leqslant lA$ и докажем, что $|F(\alpha) - F(\beta)| \leqslant lA$. Рассмотрим множество наборов $\{\alpha^0, \ldots, \alpha^l\}$, где

$$\alpha^0 = \alpha, \alpha^l = \beta, \alpha_i^j = \alpha_i + j \cdot \lceil \frac{\beta_i - \alpha_i}{l} \rceil, i = 1, \ldots, m, j = 1, \ldots, l-1.$$

Учитывая, что $\forall j = 1, \ldots, l \max\limits_{i=1,\ldots,m} |\alpha_i^j - \alpha_i^{j-1}| \leqslant A$ и $f \in R_k^A$, имеем:

$$|f(\beta) - f(\alpha)| = |\sum_{j=1}^{l}(f(\alpha^j) - f(\alpha^{j-1}))| \leqslant \sum_{j=1}^{l}|f(\alpha^j) - f(\alpha^{j-1})| \leqslant lA,$$

что и требовалось доказать.

2. Рассмотрим функцию $f(x) = A \cdot \lfloor \frac{x}{A} \rfloor$. С одной стороны, $f \in R_k^A$. С другой стороны, рассмотрим точки $x = A - 1$, $y = 2A$. Для них имеем: $|y - x| = A + 1 \leqslant B$, но $|f(x) - f(y)| = 2A > B$, откуда следует, что $f \notin R_k^A$. Свойство 2 доказано.

Лемма 2. $\exists g(x) \in R_k^{A+1}$ *($(A+1)$-устойчивая функция от одной переменной) такая, что задача проверки A-устойчивости для функций, заданных СФЭ над множеством* $\{g\} \cup R_k^1(2)$ *(функция g и все 1-устойчивые функции от двух переменных), со-NP-полна.*

Доказательство. Принадлежность задачи классу co-NP очевидна: сертификатом будет служить пара наборов значений переменных α и β из E_k^N , таких что $\max\limits_{i=1,\ldots,m} |\alpha_i - \beta_i| \leqslant A$, но $|F(\alpha) - F(\beta)| > A$.

Для доказательства NP-трудности нашей задачи сведём к ней классическую NP-полную задачу РАСКРАШИВАЕМОСТЬ ГРАФА [6]. Рассмотрим произвольный простой граф $G = (V, E)$, $V = \{v_1, \ldots, v_N\}$ — множество его вершин. Сопоставим ему СФЭ следующего вида (рис. 1):

- Все элементы схемы имеют по два входа.
- Элементы «&» вычисляют $\min(x_1, x_2)$
- Элементы «$\neq$» принимают значение 0, если оба аргумента принимают равные значения, и 1 в противном случае.
- Переменные x_i и x_j являются входами одного функционального элемента тогда и только тогда, когда в графе G вершины v_i и v_j соединены ребром.

Эта схема содержит $|V| + 2|E|$ элементов. Реализуемую ей функцию k-значной логики обозначим $f(x_1, \ldots, x_N)$. Заметим, что так как все элементы схемы вычисляют 1-устойчивые функции, то функция f также 1-устойчива.

Рассмотрим функцию от одной переменной $g(x) = \begin{cases} A+1, x = 1; \\ 0, x > 1. \end{cases}$
Эта функция очевидным образом $(A + 1)$-устойчива, а функция

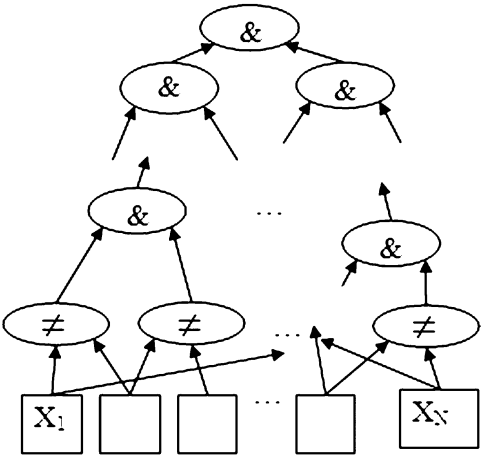

Рис. 1. СФЭ, решающая задачу о раскрашиваемости графа.

$F(x_1, \ldots, x_N) = g(f(x_1, \ldots, x_N))$ A-устойчива тогда и только тогда, когда граф G не раскрашиваем в k цветов: в этом случае $F(x_1, \ldots, x_N) \equiv 0$, а в противном случае $F(x_1, \ldots, x_N) = 0$ и существует набор из нулей и единиц, на котором F принимает значение $(A + 1)$. Теорема доказана.

Замечание. Это же доказательство без изменений можно применить для доказательства аналогичного свойства для функций, заданных *формулой*.

Доказательство теоремы 3. Реализуем функцию $g(x)$ из леммы 2 в виде формулы (а значит и СФЭ) над множеством $\{f\} \cup R_k^A(2) \cup R_k^A(1)$.

Итак, $\exists \alpha, \beta \in E_k^N : \max\limits_{i=1,\ldots,m} |\alpha_i - \beta_i| \leqslant A, |F(\alpha) - F(\beta)| > A$. Без ограничения общности считаем, что $f(\alpha) < f(\beta)$.

Рассмотрим функции $y_i(x) = \begin{cases} \beta_i, x = 1; \\ \alpha_i, x > 1. \end{cases}$ $i = 1, \ldots, m$ и функцию $h(x) = \begin{cases} 0, x \leqslant f(\alpha); \\ A, f(\alpha) < x < f(\beta); \\ A + 1, x \geqslant f(\beta) \end{cases}$

Все эти функции A-устойчивы и $g(x) = h(f(y_1(x), \ldots, y_m(x)))$. Отсюда и из свойства 1 классов R_k^A (Лемма 1) следует, что

$$g(x) \cup R_k^1(2) \subset [\{f\} \cup R_k^A(2) \cup R_k^A(1)].$$

Значит, можно применить лемму 2. Теорема доказана.

2.4. Полиномиально разрешимый случай

Для доказательства теоремы 4 приведём алгоритм, решающий задачу, докажем его корректность и оценим время работы.

Алгоритм 1.

Дано: $G = (V, E)$ — ориентированное дерево, $L \subset V$ — множество листьев, $\Phi\colon V \backslash L \to P_k$ — операторы агрегирования, соответствующие неконцевым вершинам.

1) Разобьем вершины графа на уровни $V = V_0 \cup V_1 \cup V_h$ в соответствии с длиной (единственного) пути до корневой вершины.
2) Сопоставим каждой вершине v множество $T(v)$ пар $(x, y) \in E_k^2$. Вычисление $T(v)$ осуществляем рекурсивно:
   а) Басис. $v \in L \Rightarrow T(x) = \{(x, y) \in E_k^2 : 0 \leqslant |x - y| \leqslant A\}$
   б) Рекурсивный шаг алгоритма — цикл по уровням V_i от $h-1$ до 0:
      - Рассмотрим все вершины из $V_i \setminus L$ — вершины уровня i, которые не являются листьями. Для каждой такой вершины p рассмотрим $V_p \in V_{i+1}$ — множество всех его непосредственных потомков.
      - Пусть $V_p = \{v_1, \ldots, v_{n_p}\}$, $\Phi(p) = f(x_1, \ldots, x_{n_p}) \in P_k(n_p)$. Тогда $T(p) = \{(f(x_1, \ldots, x_{n_p}), f(y_1, \ldots, y_{n_p})) : (x_1, y_1) \in Y(v_1), \ldots, (x_{n_p}, y_{n_p}) \in Y(v_{n_p})\}$
3) На выходе алгоритма получаем множество $T(s)$, где s — корневая вершина.

Лемма 3 (корректность алгоритма). *Функция $F(x_1, \ldots, x_N)$ устойчива тогда и только тогда, когда $T(s)$ не содержит пары (x, y) такой, что $|x - y| > A$.*

Доказательство. Индукцией по уровню вершины (от h до 1) докажем, что для любой вершины p множество $T(p)$ содержит все пары значений, которые можно получить на выходе вершины p, подавая на вход всевозможные пары наборов α и β, такие что $\max\limits_i |\alpha_i - \beta_i| \leqslant A$. Доказываемое утверждение вытекает отсюда непосредственно.

1) *Басис.* $p \in V_h$. Вершина является концевой, утверждение для неё очевидно.
2) *Индуктивный переход.* $p \in V_i, i < h$. Вершина p либо является концевой (тогда утверждение для неё снова очевидно), либо имеет множество непосредственных потомков $V_p = \{v_1, \ldots, v_{n_p}\}$ и присвоенную функцию $\Phi(p) = f(x_1, \ldots, x_{n_p}) \in P_k(n_p)$. По построению $T(p)$ (осуществляющему полный перебор всех комбинаций значений переменных) и предположению индукции получаем требуемое утверждение.

Лемма доказана.

Лемма 4 (время работы алгоритма). *Время работы алгоритма не превосходит* $C(n) \cdot |V|$.

Доказательство. Время работы шага 1 алгоритма — $C_1 \cdot |V|$. На шаге 2 каждая неконцевая вершина просматривается один раз, причём время, тратящееся алгоритмом на каждую вершину, ограничено величиной, зависящей только от n.

3. Задача оптимального распределения ресурсов

В отличие от задачи проверки устойчивости, решаемой на стадии разработки модели, данная задача возникает в ходе эксплуатации системы. В случае возможности влияния на некоторые параметры наблюдаемой проблемы или процесса, системы информационного мониторинга способны не только прогнозировать результат изменений значений параметров (решать прямую задачу), но и находить оптимальный (с точки зрения соотношения вложенных средств и величины достигнутых изменений) вектор изменения параметров — вектор управления (решать обратную задачу).

Ранние версии систем информационного мониторинга для этой цели использовали поиск «критического пути» — элемента модели, изменение значения которого (с сохранением значений всех остальных элементов) вызовет наибольшие изменения в состоянии всей проблемы. Однако во многих ситуациях такой подход не приводит к результату. Примером может служить схема, изображенная на рисунке

(рис. 2): увеличение значения любого единственного параметра не окажет влияние на результат — необходимо увеличить значения всех факторов одновременно.

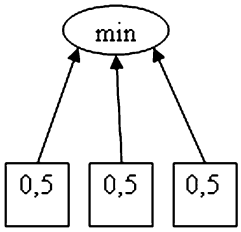

Рис. 2. Пример ситуации, при которой в схеме нет «критических путей».

Предлагаемый нами подход лишен этого недостатка — в рассматриваемом дискретном случае он всегда позволяет найти оптимальное решение. Сохраняя связь с предыдущим подходом, будем называть множество факторов, совместное изменение которых оказывает наибольшее влияние на результат, «критическим множеством».

При таком обобщении необходимо также учесть различную природу ресурсов, требуемых для изменения значений параметров. Например, если в качестве ресурса выступают денежные средства, общая стоимость вектора управления будет вычисляться как сумма стоимостей его компонент; если же ресурсом является время и изменения могут осуществляться одновременно, общей стоимостью будет максимум стоимостей.

3.1. Определения и формулировка результатов

Итак, формальная постановка задачи выглядит следующим образом: $F(x_1, \ldots, x_N)$ — функция k-значной логики, заданная схемой функциональных элементов над базисом, состоящим из всех функций от n и менее переменных, $a_1, \ldots, a_N$ — начальные значения переменных, $C_i(x)$ — стоимость присвоения i-ой переменной значения x (из текущего состояния). Для заданного C необходимо максимизировать $F(x_1, \ldots, x_N)$ при ограничении $\mathop{\perp}\limits_{i} C_i(x_i) \leqslant C$, где $\perp$ — бинарная операция, удовлетворяющая аксиомам коммутативности, ассоциативности

и монотонного неубывания, которую мы будем называть функционалом стоимости.

В общем случае NP-полнота этой задачи очевидна для любого функционала стоимости, так как NP-полна даже задача выполнимости — задача о существовании набора значений аргументов (вне зависимости от его стоимости), при котором функция принимает значение, отличное от исходного. При фиксации же функционала стоимости могут возникать частные случаи, допускающие решение за полиномиальное время. С другой стороны, в некоторых случаях даже при внесении жестких ограничений на базис функциональной схемы задача останется NP-полной. Эти ситуации мы продемонстрируем на примере двух наиболее часто встречающихся функционалах стоимости — сумме и максимуме.

Теорема 5. *Если $\perp(x, y) = \max(x, y)$ а базис СФЭ содержит только монотонные функции, то задача разрешима за линейное (от сложности схемы) время.*

Теорема 6. *Если $\perp(x, y) = x + y$, то задача оптимального распределения ресурсов NP-полна в следующей постановке:*

$F(x_1, \ldots, x_n)$ — функция алгебры логики ($k = 2$), заданная схемой функциональных элементов над множеством $\{\&, \vee\}$ (отсюда автоматически следует, что $F(0, \ldots, 0) = 0$). Для заданного натурального C необходимо проверить, существует ли такой набор $\alpha \in E_2^n, |\alpha| \leqslant C$, такой что $F(\alpha) = 1$. (по определению, $|\alpha| = \sum\limits_{i=1}^{N} \alpha_i$).

Следствие 6.1. *Задача NP-полна в следующей постановке:*
$F(x_1, \ldots, x_N)$ — функция k-значной логики ($k \geqslant 2$), заданная схемой функциональных элементов над множеством, содержащим функции от 2 переменных min *и* max. *Для заданного натурального C необходимо проверить, существует ли такой набор $\alpha \in E_k^N, |\alpha| \leqslant C$, такой что $F(\alpha) > 0$.*

Следствие 6.2. *В формулировке предыдущего следствия задача NP-полна и для функционала стоимости $\perp(x, y) = xy$.*

Труднорешаемость задачи в общем случае приводит нас к необходимости выделения частных случаев, допускающих решение эффективными алгоритмами. Некоторые такие случаи были описаны выше.

Здесь мы опишем один случаи, не зависящий от выбора функционала стоимости.

Теорема 7. *Для СФЭ, граф которых является деревом, существует алгоритм, решающий задачу оптимального распределения ресурсов за время $C(n)|V|$, где $|V|$ — число вершин, $C(n)$ — величина, зависящая только от n — максимального числа переменных базисной функции.*

Теорема 8. *Если все элементы СФЭ вычисляют линейные функции, то задача распределения ресурсов разрешима за полиномиальное время при любом функционале стоимости.*

3.2. Труднорешаемость

Доказательство теоремы 6. Принадлежность задачи классу NP очевидна: сертификатом будет служить набор α, удовлетворяющий условиям задачи. Для доказательства NP-трудности нашей задачи сведем к ней классическую NP-полную задачу МИНИМАЛЬНОЕ ПОКРЫТИЕ [6]. Нам дано множество элементов $Y = \{\hat{y}_1, \ldots, \hat{y}_M\}$, $X = \{\hat{x}_1, \ldots, \hat{x}_N\}$ — некоторое множество его подмножеств, и натуральное C. Необходимо проверить, можно ли выбрать $l \leqslant C$ элементов $\hat{x}_{i_1}, \ldots, \hat{x}_{i_l} \in X$ так, чтобы $\bigcup\limits_j \hat{x}_{i_j} = Y$.

Данным Y и X сопоставим следующую СФЭ (рис. 3):

Вплоть до предпоследнего слоя схема представляет собой бинарное дерево, все функциональные элементы вычисляют конъюнкции двух переменных. Элементы последнего слоя $y_1, \ldots, y_M$ вычисляют дизъюнкции некоторого числа переменных. Переменная x_i соединена с элементом y_j тогда и только тогда, когда $\hat{x}_i \in \hat{y}_j$.

Функцию, реализуемую этой СФЭ, обозначим $F(x_1, \ldots, x_N)$. Нетрудно проверить, что нахождение вектора α, удовлетворяющего условиям задачи, решает и задачу о наименьшем покрытии (искомые индексы $i_1, \ldots, i_l$ — номера его ненулевых компонент). Теорема доказана.

3.3. Полиномиально разрешимые случаи

Доказательство теоремы 5. Алгоритм, решающий задачу в этом случае, выглядит следующим образом:

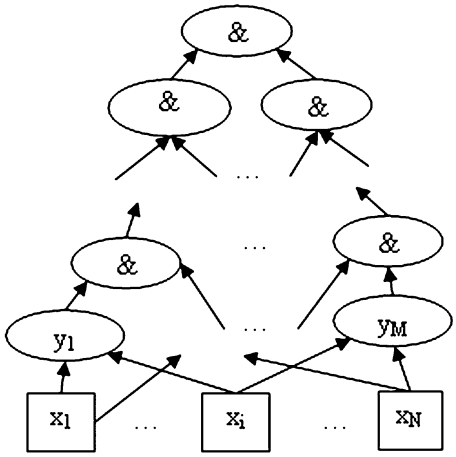

Рис. 3. СФЭ, решающая задачу о минимальном покрытии.

1) Для каждой переменной x_i находится максимальное значение a_i, такое что $C_i(a_i) \leqslant C$.
2) Вычисляется значение $F(a_1, \ldots, a_n)$. В силу монотонности F оно и является искомым максимумом.

Число шагов алгоритма на шаге 1 пропорционально числу переменных, которое не превосходит сложность схемы. На шаге 2 число шагов пропорционально сложности схемы. Теорема доказана.

Для доказательства теоремы 7 приведём алгоритм, решающий задачу, и оценим время его работы.

Алгоритм 2.

Дано: $G = (V, E)$ — ориентированное дерево, $L = \{l_1, \ldots, l_N\}$ — множество листьев, $\Phi : V \setminus L \to P_k$ — операторы агрегирования, соответствующие неконцевым вершинам, $\Psi : L \times E_k \to \mathbb{R}^+$ — «стоимости» изменения значений переменных.

1) Разобьем вершины графа на уровни $V = V_0 \cup V_1 \cup \ldots \cup V_h$ в соответствии с длиной (единственного) пути до корневой вершины.
2) Двигаясь по уровням i от h до 1, доопределим Φ на вершины из $V \setminus L$ следующим образом:

Рассмотрим все вершины из $V_i \setminus L$ — вершины уровня i, которые не являются листьями. Для каждой такой вершины p рассмотрим $V_p = \{v_1, \ldots, v_{n_p}\} \subset V_{i+1}$ — множество всех его непосредственных потомков.
Пусть $\Phi(p) = f_p(u_1, \ldots, u_{n_p}) \in P_k(n_p)$. Тогда для всех $y \in E_k$ вычисляем $\Psi(p, y) = \min\limits_{\substack{(\alpha_1,\ldots,\alpha_{n_p}):\\ f_p(\alpha_1,\ldots,\alpha_{n_p})=y}} \{\perp_{j=1}^{n_p} \Psi(v_j, \alpha_j)\}$.

3) На выходе алгоритма получаем функцию $\hat{\Psi}(y) = \Psi(s, y)$ — минимальные стоимости присвоения выходной функции значения y.

Лемма 5 (время работы алгоритма). *Время работы алгоритма не превосходит $C(n)|V|$.*

Доказательство. Время работы шага 1 алгоритма — $C_1|V|$ (очевидно). На шаге 2 каждая неконцевая вершина просматривается один раз, причем время, тратящееся алгоритмом на каждую вершину, ограничено величиной, зависящей только от n.

Доказательство теоремы 8.

Этот случай несложен и малоприменим на практике, поэтому мы ограничимся описанием лишь основных шагов алгоритма.

1) Вычисляем явный вид результирующей функции

$$F(x_1, \ldots, x_n) = \sum_{i=1}^{N} c_i x_i.$$

2) Реализуем $F(x_1, \ldots, x_n)$ СФЭ, граф которой является бинарным деревом (все функциональные элементы новой схемы вычисляют линейные функции от двух переменных).
3) Применяем алгоритм 2.

4. Более общие полиномиально разрешимые случаи для обеих задач

Алгоритмы, применимые в случае, когда граф СФЭ является деревом, можно обобщить на более общий случай. Сначала введём необходимые определения.

4.1. Определения и формулировка основных результатов

Вершина d называется *доминатором* вершины v, если d лежит на каждом пути от вершины v к корневой вершине [8].

Вершину d назовём *сверхдоминатором*, если она является доминатором для всех своих (необязательно непосредственных) потомков. Концевые вершины также будем считать сверхдоминаторами по определению.

Сверхдоминатор d называется *сверхдоминатором верхнего уровня*, если он не имеет доминаторов, являющихся сверхдоминаторами, за исключением корневой вершины.

Пусть d — сверхдоминатор. Обозначим $F_d(u_1, \ldots, u_{N_d})$ функцию k-значной логики, реализуемую на выходе вершины d. Множество переменных $\{u_1, \ldots, u_{N_d}\}$ — некоторое подножество $\{x_1, \ldots, x_N\}$. Соответствующий этой схеме подграф (состоящий из d и всех подчинённых ей вершин, а также соединяющих их рёбер) обозначим $G_d = (V_d, E_d)$. Этот подграф также будет удовлетворять всем ограничениям, перечисленным в разделе 1.2.

Пусть d — сверхдоминатор, $D_d = \{d_1, \ldots, d_{M_d}\}$ — все сверхдоминаторы верхнего уровня подграфа G_d. Обозначим $H_d(y_1, \ldots, y_{M_d})$ функцию k-значной логики, реализуемую схемой, полученной из исходной следующим образом: выходом схемы является вершина d, а входами — вершины $d_1, \ldots, d_{M_d}$ (все вершины, подчинённые вершинам-сверхдоминаторам верхнего уровня, удаляются). Соответствующий этой схеме подграф обозначим $\hat{G}_d = (\hat{V}_d, \hat{E}_d)$.

Заметим, что выполняется функциональное соотношение $F(x_1, \ldots, x_N) = H_d(F_{d_1}(X_1), \ldots, F_{d_{M_d}}(X_{M_d}))$, где $X_1, \ldots, X_{M_d}$ — некоторые упорядоченные подмножества множества $X = \{x_1, \ldots, x_N\}$, причём $\bigcup\limits_{i=1}^{M_d} X_i = X$, $X_i \cap X_j = \varnothing$, $i \neq j$.

Замечание. Похожее, но не тождественное сверхдоминатору понятие вершины интервала рассматривается в работе [5].

Уровнем вершины v назовём максимальную длину ориентированного пути от v к s (так как мы рассматриваем ациклические графы, определение корректно).

Основными результатами раздела являются следующие теоремы.

Теорема 9. *Существует алгоритм, находящий все сверхдоминаторы в произвольном ориентированном ациклическом графе с единственной корневой вершиной. Время работы алгоритма в общем случае составляет $O(p^2)$ операций (p — число вершин в графе). Алгоритм работает за время $O(p)$, когда все рёбра графа имеют ограниченную высоту (высотой ребра назовём модуль разности уровней соединяемых им вершин), и когда мощность уровней V_i возрастает экспоненциально.*

Доказательство теоремы приведено в разделе 4.2.

Теорема 10. *Пусть дана схема функциональных элементов, G — её граф, p — число вершин графа, D — множество всех сверхдоминаторов, M — максимальное число концевых вершин в подграфах $\hat{G}_d, d \in D$. Тогда существуют алгоритмы решения задач проверки устойчивости и распределения ресурсов, время работы которых не превосходит $C(n) \cdot p \cdot k^M$ для некоторой величины $C(n)$, зависящей только от n — максимальной полустепени захода вершин графа.*

Следствие 10.1. *Если $M < C \log p$ для некоторой константы C, то алгоритмы работают за полиномиальное (от числа вершин графа) время.*

Доказательство теоремы приведено в разделе 4.3.

4.2. Алгоритм поиска сверхдоминаторов

Для доказательства теоремы 9 приведём алгоритм нахождения всех сверхдоминаторов, докажем его корректность и оценим время его работы.

Алгоритм 3 (нахождения всех сверхдоминаторов).

Дано: $G = (V, E)$ — ориентированный ациклический граф, s — корневая вершина

В процессе работы алгоритм будет работать с подмножеством вершин графа — активным множеством A. В начале работы алгоритма оно является пустым. Подграф графа G, индуцированный подмножеством вершин A, обозначим G_A. Также алгоритм будет производить окраску вершин — присвоение вершинам элементов множества $\{C_1, \ldots, C_{|V|}\}$.

1) Разобьём вершины графа на уровни $V = V_0 \cup V_1 \cup \ldots \cup V_h$ (вершины из V_j будем называть вершинами уровня j) следующим образом:
   а) $V_0 = \{s\}$
   б) Если u — непосредственный потомок $v \in V_j$, то u присваивается уровень $j + 1$ (если u уже имело уровень i, то ей присваивается уровень $\max(i, j + 1)$).
2) Раскрашиваем все вершины уровня V_h в разные цвета, добавляем их в активное множество и помечаем как сверхдоминаторы.
3) Цикл по уровням i от h до 1:
   а) Добавляем в активное множество все вершины уровня $i - 1$, а также все вершины, являющиеся непосредственными родителями вершин уровня i.
   б) Перекрашиваем вершины активного множества в максимальное число цветов с соблюдением следующих свойств:
      • Вершины, имевшие один цвет в предыдущей раскраске, получают один цвет.
      • Вершины, соединённые ребром, получают один цвет.

      Для этого:
      – Для каждого цвета выбираем произвольную вершину этого цвета и соединяем её (помеченными) ребрами с остальными вершинами этого цвета.
      – Каждой компоненте связности G_A (в слабом смысле, то есть без учёта ориентации рёбер) присваиваем свой цвет.
      – Удаляем все введённые на шаге (i) рёбра.
   в) Удаляем из активного множества все вершины уровня i.
   г) Все вершины, являющиеся единственными носителями своего цвета и имеющие уровень $i - 1$, помечаем как сверхдоминаторы.

   Конец цикла

Корректность работы алгоритма обосновывается двумя утверждениями:

Лемма 6. *Любая вершина, являющаяся сверхдоминатором, будет помечена алгоритмом.*

Доказательство. Рассмотрим произвольный сверхдоминатор d (пусть он имеет уровень i_d) и соответствующий ему подграф G_d. Удалим из графа G все рёбра, исходящие из d. По определению сверхдоминатора, после этой операции граф G становится несвязным и граф G_d образует его компоненту связности. Для завершения доказательства заметим, что с одной стороны, эта операция не влияет на ход выполнения алгоритма на итерациях вплоть до уровня $i_d + 1$ включительно, а с другой стороны, в модифицированном графе G вершина d будет помечена алгоритмом на уровне $i_d + 1$, так как она является единственной вершиной уровня i_d в компоненте связности G_d.

Лемма 7. *Любая вершина, не являющаяся сверхдоминатором, не будет помечена алгоритмом.*

Доказательство. Рассмотрим произвольную вершину, не являющуюся сверхдоминатором, и обозначим её через u. По условию и определению сверхдоминатора, у неё существует потомок v, соединённый с корневой вершиной s путём, не проходящим через u. Найдём на этом пути вершину w, имеющую наибольший уровень, не превосходящий уровень вершины u, и заметим, что при окраске алгоритмом вершины u и w получат одинаковый цвет. Тем самым u не будет помечена как сверхдоминатор.

Лемма 8. *Время работы алгоритма не превышает $O(N^2)$ операций.*

Доказательство. Сначала оценим время, тратящееся алгоритмом за одну итерацию цикла. Итак, пусть граф G_A имеет N_A вершин и, в силу ограничения на полустепень захода вершины (см. раздел 1.2), не более $n \cdot N_A$ рёбер. Тогда:

i Добавление рёбер займёт $O(N_A)$ шагов и добавит не более N_A новых рёбер.

ii Раскраска связных компонент графа заключается в обходе всех его рёбер, поэтому она займёт $O(N_A)$ шагов.

iii Удаление рёбер займёт $O(N_A)$ шагов.

Итак, на одну итерацию цикла алгоритма тратится время, пропорциональное мощности активного множества. Это означает, что

общее время работы алгоритма не превышает $O(N^2)$ операций (затраты времени на шагах 1 и 2 очевидным образом линейны), что и требовалось доказать.

В общем случае эту оценку для данного алгоритма нельзя улучшить. Примером может служить граф $G = (V, E)$, у которого $V = 1, 2, \ldots, p$, а $E = \{(i+1, i), i = 1, \ldots, p-1\} \cup \{(p, i), i+1, \ldots, p-1\}$.

Однако можно сформулировать дополнительные условия на вид графа, которые часто выполняются на практике и которые обеспечивают быструю (за линейное время) работу алгоритма.

Лемма 9 (Условие 1). *Назовём высотой ребра модуль разности уровней соединяемых им вершин. Тогда если все рёбра графа имеет ограниченную (величиной, не зависящей от p — числа вершин в графе) высоту, то алгоритм работает за линейное время.*

Доказательство. Обозначим максимальную высоту ребра через m. Тогда любая вершина графа может находиться в активном множестве не более m раз. Обозначив через T_i время, тратящееся алгоритмом при выполнении итерации цикла на уровне i, а через N_{A_i} — соответствующую мощность активного множества, получим, учитывая линейную зависимость времени на итерацию от текущей мощности активного множества: $T_i \leqslant cN_{A_i}$, следовательно, $T = \sum\limits_{i=0}^{k} T_i \leqslant c \sum\limits_{i=0}^{k} N_{A_i} \leqslant c \cdot m \cdot p$. Лемма доказана.

Лемма 10 (Условие 2). *Пусть мощность уровней V_i возрастает экспоненциально, то есть $\exists \alpha > 1 : \forall i = 0, \ldots, h-1 : |V_{i+1}| > \alpha |V_i|$. Тогда алгоритм работает за линейное время.*

Доказательство. Обозначим $p_i = |V_i|$ — мощность i-го уровня, $P_i = \sum\limits_{j=0} ip_j$ — число вершин в уровнях от 0 до i. По условию, $p_i \leqslant \frac{p_h}{\alpha^{h-i}}$, следовательно, $P_i = \sum\limits_{j=0}^{i} p_j \leqslant p_i \sum\limits_{j=0}^{i} \frac{1}{\alpha^{i-j}} < \frac{p_h}{\alpha^{h-j}} \sum\limits_{j=0}^{\infty} \frac{1}{\alpha^k} = \frac{p_h}{\alpha^{h-i-1}(\alpha-1)} = c_1 \frac{p_h}{\alpha^{h-i}}$, где $c_1 = \frac{\alpha}{\alpha-1}$.

Обозначив за T_i время, тратящееся алгоритмом при выполнении итерации цикла на уровне i, получим, учитывая линейную зависимость времени на итерацию от текущей мощности активного множе-

ства: $T_i \leqslant c_2 P_i$, следовательно, $T = \sum\limits_{i=0}^{h} T_i \leqslant c_1 c_2 p_h \sum\limits_{i=0}^{h} \alpha^{i-h} < c_1^2 c_2 p_h < c_1^2 c_2 p$. Лемма доказана.

4.3. Алгоритмы решения задач проверки устойчивости и оптимального распределения ресурсов

Теперь, имея алгоритм поиска сверхдоминаторов, осуществляющий разделение нашей схемы функциональных элементов на функционально-независимые подсхемы, используем его для решения наших задач в общем случае.

Алгоритм 4. ***(Рекурсивный алгоритм решения задачи проверки устойчивости)***

Дано: $G = (V, E)$ — ориентированный ациклический граф, $L \subset V$ — множество листьев, $D \subset V$ — множество сверхдоминаторов, s — корневая вершина, $\Phi\colon V \setminus L \to P_k$ — операторы агрегирования, соответствующие неконцевым вершинам.

Алгоритм сопоставялет вершине s множество $T(s)$ пар $(x, y) \in E_K^2$ следующим рекурсивным образом:

1) Если корневая вершина s является концевой (то есть граф состоит из одной вершины), $T(s) = \{(x, y) \in E_K^2 : 0 \leqslant |x - y| \leqslant A\}$. В противном случае:
2) Находим в множестве D подмножество $D' = \{d_1, \ldots, d_M\}$ всех сверхдоминаторов верхнего уровня.
3) Рекурсивно вычисляем $T(d)$ для всех $d \in D'$.
4) Полагаем $T(s) = \{(H_s(x_1, \ldots, x_M), (H_s(y_1, \ldots, y_M)) : (x_1, y_1) \in T(d_1), \ldots, (x_M, y_M) \in T(d_M)\}$.

Как и в «деревянном» случае, функция $F(x_1, \ldots, x_N)$ устойчива тогда и только тогда, когда $T(s)$ не содержит пары (x, y) такой, что $|x - y| > A$.

Алгоритм 5. ***(Рекурсивный алгоритм решения задачи управления)***

Дано: $G = (V, E)$ — ориентированный ациклический граф, $L \subset V$ — множество листьев, $D \subset V$ — множество сверхдоминаторов, s — корневая вершина, $\Phi\colon V \setminus L \to P_k$ — операторы агрегирования, соответствующие неконцевым вершинам, $\Psi\colon L \times E_k \to \mathbb{R}^+$ — «стоимости» изменения значений переменных.

Алгоритм рекурсивно доопределяет функцию Ψ для всех вершин-сверхдоминаторов:

1) Если корневая вершина s является концевой (то есть граф состоит из одной вершины), то $\Psi(s, \cdot)$ определена условием.
   В противном случае:
2) Находим в множестве D подмножество $D' = \{d_1, \ldots, d_M\}$ всех сверхдоминаторов верхнего уровня.
3) Рекурсивно вычисляем $\Psi(s, \cdot)$ для всех $d \in D'$.
4) $\Psi(s, y) := \min\limits_{\substack{(\alpha_1,\ldots,\alpha_M):\\ H_s(\alpha_1,\ldots,\alpha_M)=y}} \left\{ \mathop{\perp}\limits_{j=1}^{M} \Psi(d_j, \alpha_j) \right\}$.

На выходе алгоритма получаем функцию $\hat{\Psi}(y) = \Psi(s, y)$ — минимальные стоимости присвоения выходной функции значения y.

Корректность работы обоих алгоритмов доказывается аналогично случаю графов-деревьев. Оценим время работы алгоритмов.

1) Нахождение всех сверхдоминаторов верхнего уровня для каждого подграфа заключается в однократном обходе всех рёбер графа, поэтому требует $O(|E|)$ шагов. В силу ограничений на граф, введённых в разделе 1.2, это время также равно $O(|V|)$.
2) Время вычисления множеств $T(d)$ (функций $\Psi(d, \cdot)$) для всех сверхдоминаторов вычислим как сумму времен работы алгоритма для каждого сверхдоминатора:
   а) $T(s) = \{(H_s(x_1, \ldots, x_M), (H_s(y_1, \ldots, y_M)) : (x_1, y_1) \in T(d_1), \ldots, (x_M, y_M) \in T(d_M)\}$
   В худшем случае для нахождения $T(s)$ нужно вычислить H_s на всех возвожных k^M наборах. Вычисление H_s на одном наборе занимает $C(n)|V_s|$ шагов. Итого, время работы для подграфа $\hat{G}_d$, имеющего p_d вершин и M_d концевых вершин — $C_1(n) \cdot p_d \cdot k^M$ шагов.

б) $\Psi(s,y) := \min\limits_{\substack{(\alpha_1,\ldots,\alpha_M):\\ H_s(\alpha_1,\ldots,\alpha_M)=y}} \left\{ \bigperp_{j=1}^{M} \Psi(d_j,\alpha_j) \right\}$

Для вычисления минимума необходимо перебрать k^M сумм. Для вычисления каждой суммы требуется выполнить $M-1$ сложений и вычислить значение $H_s(x_1,\ldots,x_M)$, что занимает $C(n)|V_s|$ шагов. Итого, время работы для подграфа $\hat{G}_d$, имеющего p_d вершин и M_d концевых вершин — $C_2(n)\cdot pd\cdot k^M$ шагов.

Заметим также, что в совокупность всех подграфов $\hat{G}_d, d\in D$ каждая вершина входит ограниченное число раз (два раза, если она является сверхдоминатором, и один раз, если не является). Поэтому $T=\sum\limits_{d\in D} T(\hat{G}_d) \leqslant \sum\limits_{d\in D} C(n)\cdot p_d\cdot k^{M_d} \leqslant C(n)\cdot k^{M_{max}} \sum\limits_{d\in D} p_d \leqslant 2C(n)\cdot k^{M_{max}}\cdot p$.

Таким образом, теорема 10 доказана.

Заметим, что время работы приведённых алгоритмов зависит только от графовой структуры схемы. Таким образом, мы имеем возможность оценить время решения задач проверки устойчивости или распределения ресурсов (которые, напомним, являются труднорешаемыми в общем случае) для конкретной схемы сразу после применения к ней алгоритма поиска сверхдоминаторов (работающего за полиномиальное время).

5. Заключение

В работе была предложена формализация технологии информационного мониторинга на основе теорий схем функциональных элементов и функций k-значной логики. В рамках этой формализации были рассмотрены две основные задачи — проверки устойчивости и оптимального распределения ресурсов. Для обеих задач была доказана их труднорешаемость в общем случае и выделены частные случаи моделей, допускающие применение быстрых алгоритмов. Также был получен алгоритм, проверяющий принадлежность произвольной модели этим частным случаям.

Дальнейшие исследования будут направлены на расширение множества полиномиально разрешимых случаев для обеих задач, использование методов сокращения перебора, исследование возможности

применения приближённых алгоритмов для задачи распределения ресурсов и другие вопросы.

Автор выражает признательность А. П. Рыжову за постановку задачи, а также В. Б. Кудрявцеву и Э. Э. Гасанову за обсуждение работы и ценные рекомендации.

Список литературы

[1] Заде Л. А. Понятие лингвистической переменной и его применение к принятию приблизительных решений. — М.: Мир, 1976.

[2] Кудрявцев В. Б. Функциональные системы. — М.: Изд-во Моск. Ун-та, 1982.

[3] Рыжов А. П. Об агрегировании информации в нечетких иерархических системах // Интеллектуальные системы. — 2002. — Т. 6. Вып. 1–4. — С. 341–364.

[4] Яблонский С. В. Основные понятия кибернетики // Проблемы кибернетики. — 1959. — Вып. 2. — С. 7–38.

[5] Cocke J. Global common subexpression elimination // ACM SIGPLAN Notices 5:7. — 1970. — P. 20–24.

[6] Karp R. M. Reducibility among combinatorial problems // Complexity of computer computations. — NY: Plenum Press, 1974. — P. 85–103.

[7] Ryjov A. Basic principles and foundations of information monitoring systems // Monitoring, Security, and Rescue Techniques in Multiagent Systems. — Springer, 2005. — P. 147–160.

[8] Tarjan R. Finding dominators in directed graphs // SIAM J Computing. — 1974. — 3, № 1. — P. 62–89.