УДК 004.02:004.047
doi: 10.21685/2587-7704-2024-9-2-7

Open Access | RESEARCH ARTICLE

# Применение логического программирования в решении задач переборного характера на графах

***Андрей Геннадьевич Михалев***
Пензенский государственный университет, Россия, г. Пенза, ул. Красная, 40
mag@pnzgu.ru

***Кирилл Александрович Жигалов***
Пензенский государственный университет, Россия, г. Пенза, ул. Красная, 40
zhigalov-05@inbox.ru

**Аннотация.** Рассматривается подход к установлению соответствия между графами на основе перебора всех возможных комбинаций соответствия между вершинами двух графов с оценкой каждой из таких комбинаций. На примере программной реализации анализируются возможности языка Prolog в решении задач переборного характера.

**Ключевые слова**: графы, соответствие графов, рекурсивная обработка, Prolog, программная реализация

**Для цитирования**: Михалев А. Г., Жигалов К. А. Применение логического программирования в решении задач переборного характера на графах // Инжиниринг и технологии. 2024. Т. 9 (2). С. 1–4. doi: 10.21685/2587-7704-2024-9-2-7

# Application of logic programming in solving search problems on graphs

***Andrey G. Mikhalev***
Penza State University, 40 Krasnaya Street, Penza, Russia
mag@pnzgu.ru

***Kirill A. Zhigalov***
Penza State University, 40 Krasnaya Street, Penza, Russia
zhigalov-05@inbox.ru

**Abstract.** An approach to establishing correspondence between graphs is considered based on enumerating all possible combinations of correspondence between the vertices of two graphs with an assessment of each of such combinations. Using the example of software implementation, the capabilities of the Prolog language in solving enumeration problems are analyzed.

**Keywords**: graphs, graph matching, recursive processing, Prolog, software implementation

**For citation**: Mikhalev A.G., Zhigalov K.A. Application of logic programming in solving search problems on graphs. *Inzhiniring i tekhnologii = Engineering and Technology*. 2024;9(2):1–4. (In Russ.). doi: 10.21685/2587-7704-2024-9-2-7

Применение графов и моделей на их основе является популярным и эффективным средством моделирования различных систем и процессов в самых разнообразных сферах: компьютерные науки, биоинформатика, социальные сети и многие другие. Одной из актуальных задач при работе с графами является установление соответствия одного графа другому. В данной работе рассматривается взгляд на решение такой задачи применительно к установлению взаимно однозначного соответствия между вершинами двух графов, которые изначально могут быть представлены в разных формах. Успешное сопоставление этих двух представлений между собой означает, что разными формами представлен один и тот же граф [1].

Первый шаг в этом процессе – приведение двух исходных представлений графов к виду, который может быть эффективно использован в программе. К примеру, имеются исходные представления графов в виде матриц смежности (рис. 1).

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| A | | 1 | 1 | 1 | | | |
| B | 1 | | 1 | | 1 | | |
| C | 1 | 1 | | | | 1 | 1 |
| D | 1 | | | | | 1 | 1 |
| E | | 1 | | | | | 1 |
| F | | | 1 | 1 | | | |
| G | | | 1 | 1 | 1 | | |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | | | | | 1 | 1 | |
| 2 | | | | 1 | 1 | | 1 |
| 3 | | | | 1 | | | 1 |
| 4 | | 1 | 1 | | 1 | 1 | |
| 5 | 1 | 1 | | 1 | | | |
| 6 | 1 | | | 1 | | | 1 |
| 7 | | 1 | 1 | | | 1 | |

Рис. 1. Исходные представления графов

В программе исходные представления графов можно задать следующими структурами (рис. 2).

| | |
|---|---|
| *connected_to(a, [b, c, d]).* | *adjacent(1, [5, 6]).* |
| *connected_to(b, [a, c, e]).* | *adjacent(2, [4, 5, 7]).* |
| *connected_to(c, [a, b, f, g]).* | *adjacent(3, [4, 7]).* |
| *connected_to(d, [a, f, g]).* | *adjacent(4, [2, 3, 5, 6]).* |
| *connected_to(e, [b, g]).* | *adjacent(5, [1, 2, 4]).* |
| *connected_to(f, [c, d]).* | *adjacent(6, [1, 4, 7]).* |
| *connected_to(g, [c, d, e]).* | *adjacent(7, [2, 3, 6]).* |

Рис. 2. Представления графов в программе

При использовании такого способа с каждой вершиной графа связывается список смежных с ней вершин.

Один из возможных подходов к решению задачи состоит в генерации всех возможных комбинаций соответствия вершин в двух исходных представлениях и проверке каждой комбинации, действительно ли она отражает необходимое соответствие всех вершин графа в одной форме представления аналогичным вершинам в другой форме представления графа.

Например, для поиска соответствия вершин двух графов набор вершин [*a, b, c, d, e, f, g*] в первом представлении (вершины именуются латинскими буквами) может быть сопоставлен с набором [1, 2, 3, 4, 5, 6, 7] во втором представлении графа (вершины именуются числами), затем с набором [1, 2, 3, 4, 5, 7, 6], затем – [1, 2, 3, 4, 6, 5, 7] и т.д.

Все возможные варианты сопоставления последовательно должны быть проверены, устанавливает ли какой-либо из них искомое взаимно однозначное соответствие между вершинами двух исходных представлений графов. При такой проверке для каждой вершины требуется убедиться, что связи между вершинами в первом представлении графа в точности соответствуют связям между аналогичными вершинами во втором представлении графа. Если проверка очередного варианта сопоставления успешна, это значит, что установлено соответствие между двумя описаниями, т.е. этими двумя описаниями задан один и тот же граф. Для графов в рассматриваемом примере такое соответствие будет представлено, например, набором пар: [(*a*,2), (*b*,5), (*c*,4), (*d*,7), (*e*,1), (*f*,3) (*g*,6)]. Это означает, что набор вершин [*a, b, c, d, e, f, g*] успешно сопоставлен с набором [2, 5, 4, 7, 1, 3, 6].

Реализация такого подхода представляется как рекурсивный процесс генерации возможных комбинаций вершин, проверки каждой комбинации с перебором ее вершин и анализом их взаимосвязей. Поэтому данные для программы целесообразно представить в виде рекурсивно определяемых структур и применить язык, ориентированный на рекурсивную обработку – *Prolog* [2, 3]. Вместе с этим, *Prolog,*

являясь представителем языков логического программирования, хорошо подходит для решения задач переборного характера, так как механизмы этого языка ориентированы не на запись алгоритма обработки исходных данных, а на описание характера задачи через описание отношений между ее объектами и величинами с последующей формулировкой запроса к программе для поиска всех возможных решений [4].

Разработанная для реализации данной задачи программа выполняет следующие действия.

Осуществляется генерация возможных комбинаций соответствия вершин в исходных представлениях графов. Для этого из описания графа в первом представлении формируется список его вершин ([*a, b, c, d, e, f, g*]); определяется длина *Len* этого списка и генерируется список чисел от 1 до *Len* (содержательно – это перечень вершин графа во втором представлении); формируется возможная перестановка элементов последнего списка; на основании полученной перестановки формируется комбинация соответствия вершин как список пар вида [(«буква», «число»), («буква», «число»), ...], представляющая собой потенциально возможный вариант решения задачи, который должен быть проверен на корректность.

Проверка на корректность каждой полученной комбинации соответствия вершин заключается в установлении факта, что для каждой вершины из одного представления графа есть соответствующая вершина в другом представлении графа. Для этого из описания графа в первом представлении формируется список смежных вершин для первой вершины графа (*a,* [*b, c, d*]); из поверяемого возможного варианта списка пар вида [(«буква», «число», …)] находятся номера этих смежных вершин [2, 3, 4]; получается список смежных вершин из второго представления графа (1, [5, 6]); после их сортировки сравниваются списки смежных вершин первого представления графа со вторым представлением графа ([2, 3, 4] = [5, 6]). Если списки совпадают, то первой вершине из первого представления графов найдена соответствующая ей вершина во втором представлении графа. В таком случае поиск соответствия вершин из второго представления продолжается рекурсивно для очередной вершины из первого представления графов. Если же соответствие не установлено, то формируется новая комбинация соответствия вершин как список пар вида [(«буква», «число»), ...] и для нее повторяются те же действия. Таким образом, будет установлено соответствие между вершинами двух представлений графов и проверено, являются ли эти представления одним и тем же графом.

В результате выполнения таких рекурсивных действий программы будет установлено, что исходными представлениями задан один и тот же граф с учетом взаимного соответствия вершин. Результат работы программы для рассматриваемого примера представлен ниже (рис. 3).

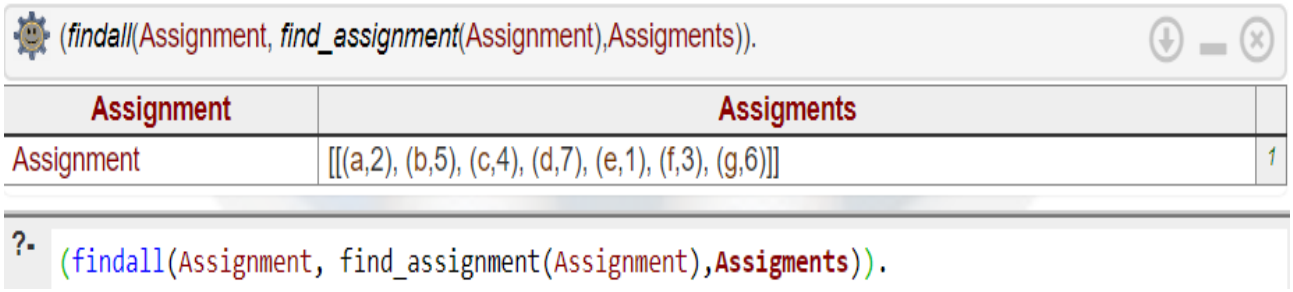


Рис. 3. Результат работы программы

Очевидными достоинствами данного решения задачи являются компактность и лаконичность программы, реализующей предложенное решение, простота понимания способа решения, возможность нахождения всех вариантов соответствия вершин (если они имеются), модульность (так как реализация такого подхода разбита на небольшие логические части). Можно отметить, что предложенное решение может быть оптимизировано, например, за счет оценки на начальных стадиях работы первичных характеристик графов – количества вершин, степеней вершин, а также за счет усовершенствования отдельных этапов решения.

Тем не менее основной недостаток использованного подхода состоит в применении полного перебора комбинаций соответствия вершин, что является весьма ресурсоемким процессом. Однако здесь существенную помощь может оказать привлечение техники табулирования, находящей использование в *SWI-Prolog*, одной из популярных версий этого языка.

Табулирование в *SWI-Prolog* заключается в сохранении результатов выполнения функций во внутренних рабочих таблицах, в которых сохраняются результаты предыдущих вычислений для различных исходных данных. И когда происходит обращение к функции с ранее встречавшимися аргументами, результат может быть быстро извлечен из таблиц, что позволяет избежать излишних

повторных вычислений. Такой подход значительно ускоряет выполнение программ, особенно реализующих интенсивные вычисления с частыми повторными вызовами функций. И особенно эффективно это в случае применения рекурсивных функций, где одни и те же вычисления могут повторяться многократно.

## Список литературы

1. Валетов В. А., Орлова А. А., Третьяков С. Д. Интеллектуальные технологии производства приборов и систем : учеб. пособие. СПб : СПб ГУИТМО, 2008. 134 с.
2. Стерлинг, Л., Шапиро Э. Искусство программирования на языке Пролог. М. : Мир, 1986. 235 с.
3. Клоксин У., Меллиш К. Программирование на языке Пролог. М. : Мир, 1987. 336 с.
4. Малпас Дж. Реляционный язык Пролог и его применение. М. : Наука, 1990. 464 с.

## References

1. Valetov V.A., Orlova A.A., Tret'jakov S.D. *Intellektual'nye tehnologii proizvodstva priborov i sistem: ucheb. posobie = Intelligent technologies for the production of devices and systems : studies. stipend. Saint Petersburg*: SPb GUITMO, 2008:134. (In Russ.)
2. Sterling L., Shapiro Je. *Iskusstvo programmirovanija na jazyke Prolog = The art of programming in the Prolog language*. Moscow: Mir, 1986:235. (In Russ.)
3. Kloksin U., Mellish K. *Programmirovanie na jazyke Prolog = Programming in the Prolog language*. Moscow: Mir, 1987:336. (In Russ.)
4. Malpas Dzh. *Reljacionnyj jazyk Prolog i ego primenenie = Relational language Prolog and its application*. Moscow: Nauka, 1990:464. (In Russ.)

**Поступила в редакцию / Received** 21.02.2024

**Принята к публикации / Accepted** 21.03.2024