Математические структуры и моделирование
2026. № 1 (77). С. 53–62

УДК 519.8
DOI 10.24147/2222-8772.2026.1.53-62

ЭВОЛЮЦИОННЫЕ АЛГОРИТМЫ ДЛЯ ОПТИМИЗАЦИОННОЙ ВЕРСИИ ЗАДАЧИ ГРАФ-ПОДГРАФ ИЗОМОРФИЗМА

Ю.В. Захарова[1, 2]
к.ф.-м.н., старший научный сотрудник, e-mail: yzakharova@ofim.oscsbras.ru
В.П. Лобанов[1, 2]
аспирант, e-mail: lobanov.vasiliy.pavlovich@gmail.com
Г.Р. Кравцов[3]
студент, e-mail: grk55rus@mail.ru

[1]Омский филиал Института математики им. С.Л. Соболева СО РАН, Омск, Россия
[2]Новосибирский государственный университет
[3]Омский государственный технический университет

Аннотация. В статье предлагаются эволюционные алгоритмы для оптимизационной версии задачи граф-подграф изоморфизма, которая имеет приложения в компьютерных системах и в биоинформатике. Генетический алгоритм использует операторы, основанные на наследовании или мутации значений в позициях перестановок, используемых для кодировки решений. Эволюционная стратегия задействует адаптивный выбор операторов. Экспериментальные исследования показывают применимость разработанных алгоритмов.

Ключевые слова: эволюционная стратегия, генетический алгоритм, изоморфизм, эксперимент.

Введение

В статье рассматривается оптимизационный вариант задачи граф-подграф изоморфизма, где по заданным двум помеченным графам (малому и большому) требуется найти «максимально похожий» на малый граф подграф в большом графе [4, 6]. Рассматриваемая задача имеет практическое приложение в различных областях. Одной из которых является биоинформатика, где сопоставление сетей (Network Alignment) используется для выявления структурных сходств между биологическими связями. Это позволяет переносить свойства функций белков и генов от хорошо изученных видов к менее изученным. Такой подход используется при анализе взаимодействия белков (PPI), выравнивании геномных последовательностей и метаболических сетях [6]. Кроме того, взвешенный вариант задачи граф-подграф изоморфизма возникает в системах распределённых вычислений в задачах рационального использования ресурсов [4]. При распределении нагрузки требуется эффективно распределять операции между вычислительными узлами, учитывая такие параметры, как интенсивность обмена данными и загрузку процессоров. Ещё одна область применения – оптимизация работы многопроцессорных и серверных систем, где

требуется размещать операции с учётом структуры связей между компонентами [4]. Аналогичные задачи возникают в транспортных, биологических и социальных сетях.

В статье [6] авторы для решения оптимизационной версии задачи граф-подграф изоморфизма разработали гибридный меметический алгоритм, сочетающий генетический алгоритм и локальный поиск. Алгоритм переборного типа представлен в [8].

В настоящей работе предлагается адаптивная эволюционная стратегия для решения задачи граф-подграф изоморфизма. В алгоритме для адаптивного выбора операторов используется обучение с подкреплением, что позволяет находить решения хорошего качества при совместной работе операторов.

1. Постановка задачи

Даны два неориентированных графа $G_1 = (V_1, E_1)$ и $G_2 = (V_2, E_2)$, где V_i – множество вершин, E_i – множество рёбер. Здесь $|V_2| \leqslant |V_1|$. Требуется найти такое сопоставление $\varphi : V_2 \to V_1$, что функция

$$f(\varphi, G_1, G_2) = \frac{1}{2} \sum_{u,v \in V_2} \tau_\varphi(u, v)$$

принимает максимальное значение, где

$$\tau_\varphi(u, v) = \begin{cases} 1, & \text{если } \{u, v\} \in E_2 \text{ и } \{\varphi(u), \varphi(v)\} \in E_1, \\ 0 & \text{иначе.} \end{cases}$$

Данная задача является NP-трудной, так как к ней полиномиально сводится задача об изоморфном подграфе (см., например, [6]).

Далее составим модель целочисленного линейного программирования (ЦЛП). Введём булевы переменные x_{ui} такие, что

$$x_{ui} = \begin{cases} 1, & \text{если } \varphi(u) = i, \\ 0, & \text{иначе,} \end{cases} \qquad u \in V_2,\ i \in V_1.$$

Также введём булевы переменные w_{uvij} такие, что

$$w_{uvij} = \begin{cases} 1, & \text{если } \varphi(u) = i \text{ и } \varphi(v) = j, \\ 0, & \text{иначе,} \end{cases} \qquad \{u, v\} \in E_2,\ \{i, j\} \in E_1.$$

Модель ЦЛП для задачи может быть записана следующим образом:

$$\sum_{\{u,v\}\in E_2}\ \sum_{\{i,j\}\in E_1} w_{uvij} \to \max, \tag{1}$$

$$\sum_{i\in V_1} x_{ui} = 1, \quad \forall u \in V_2, \tag{2}$$

$$\sum_{u\in V_2} x_{ui} \leqslant 1, \quad \forall i \in V_1, \tag{3}$$

$$w_{uvij} \leqslant x_{ui}, \quad \forall\{u,v\} \in E_2,\ \forall\{i,j\} \in E_1, \tag{4}$$

$$w_{uvij} \leqslant x_{vj}, \quad \forall\{u,v\} \in E_2,\ \forall\{i,j\} \in E_1, \tag{5}$$

$$w_{uvij} \geqslant x_{ui} + x_{vj} - 1, \quad \forall\{u,v\} \in E_2,\ \forall\{i,j\} \in E_1, \tag{6}$$

$$x_{ui} \in \{0,1\}, \quad w_{uvij} \in \{0,1\}, \quad \forall u,\ v \in V_2,\ \forall i,\ j \in V_1. \tag{7}$$

Целевая функция (1) максимизирует число рёбер графа G_2, которые при сопоставлении φ сохраняются, то есть переходят в рёбра графа G_1. Ограничение (2) гарантирует, что отображение φ определено для всех вершин G_2: каждая вершина выбирает ровно один образ. Ограничение (3) обеспечивает, что разные вершины G_2 не могут быть сопоставлены одной и той же вершине G_1. Ограничения (4), (5), (6) эквивалентны условию $w_{uvij} = x_{ui} \cdot x_{vj}$, но записаны в линейном виде.

2. Эволюционные алгоритмы

Для поиска приближенного решения задачи предлагается эволюционная стратегия $(1+1)$-ES и генетический алгоритм, которые представляют собой эвристический метод оптимизации из класса эволюционных алгоритмов, основанный на адаптации и эволюции особей (пробных решений) [3, 7].

В качестве кодировки решений используется перестановка. К множеству вершин малого графа добавляются изолированные вершины, чтобы количество вершин в двух графах было одинаковым. Позиции перестановки соответствуют номерам вершин графа G_1, а значения в позициях – номерам вершин графа G_2.

При вычислении целевой функции сопоставляются пары вершин и осуществляется проверка наличия ребра в графах. Если ребро есть в обоих графах, то целевая функция увеличивается на 1, иначе – остаётся с прежним значением.

Эволюционная стратегия. Работа алгоритма начинается с особи-перестановки, сгенерированной случайным образом. На каждой итерации к текущей перестановке применяется один из операторов мутации, вносящих случайные изменения. Если действие оператора приводит к перестановке, соответствующей улучшению значения целевой функции по сравнению с текущим решением, то на следующей итерации используется построенная перестановка. В противном случае работа продолжается с текущей перестановкой. Схема данного алгоритма представлена ниже.

Эволюционная стратегия $(1+1)$-ES.

1. Построить начальную перестановку π.
2. Пока не выполнен критерий остановки:
    1. Выбрать оператор мутации *mut*.

2. Построить $\pi' = mut(\pi)$.
3. Обновить π как лучшую по целевой функции перестановку между π и π'.

В $(1 + 1)$-ES используются три различных оператора мутации: swap, shift и shuffle [7]. Первый оператор меняет местами два случайных элемента перестановки, второй оператор сдвигает случайный элемент перестановки в новую случайную позицию, третий же выбирает случайную подпоследовательность и случайным образом перемешивает элементы этой подпоследовательности. Для выбора оператора на каждой итерации алгоритма используется метод на основе обучения с подкреплением [1], основанного на целенаправленном обучении, которое производится путём взаимодействия условного агента с некоторой средой, в процессе которого направление действий агента обновляется на основе полученного опыта (см., рис. 1). Последовательность действий агента с вероятностью их выбора в некотором состоянии формируют стратегию агента. Целью агента является максимизация суммарного вознаграждения, которое он получает во время взаимодействия со средой.

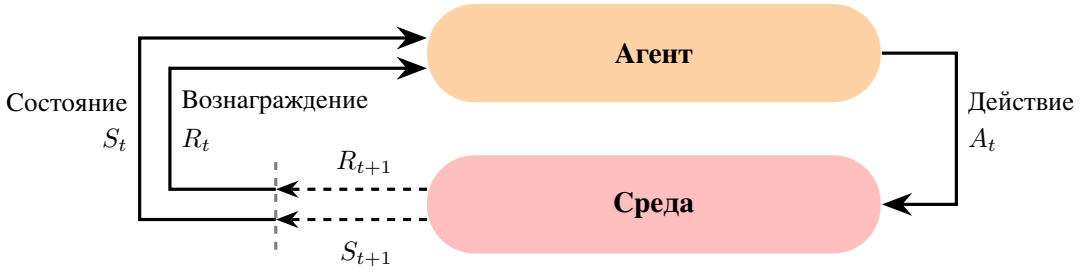

Рис. 1. Обучение с подкреплением

Существуют, например, такие алгоритмы обучения с подкреплением, как SARSA, Q-learning, метод на основе решения задачи о многоруком бандите и другие [1].

В данной работе используется метод обучения с подкреплением для выбора оператора мутаций. Адаптивный выбор операторов состоит из двух стадий:

1. Стадия обучения

Операторы: $op_1, op_2, op_3, \ldots, op_k$.

a) Операторы выбираются равновероятно.

b) Обновление веса оператора $op_i,\ i = 1, \ldots, k$:

$$w_i = \lambda \cdot \text{score} + (1 - \lambda) \cdot w_i,$$

где λ – параметр схемы, *score* – награда за улучшение (ухудшение) значения целевой функции.

2. Стадия подкрепления

a) Оператор $op_i,\ i = 1, \ldots, k$ выбирается с вероятностью:

$$p_i = \frac{w_i}{\sum_{j=1}^{k} w_j},$$

b) Обновление веса оператора $op_i,\ i = 1, \ldots, k$:

$$w_i = \lambda \cdot score + (1 - \lambda) \cdot w_i.$$

Генетический алгоритм. Для решения задачи также предлагается генетический алгоритм со стационарной схемой воспроизведения, где решения как и в случае эволюционной стратегии кодируются перестановками.

В качестве оператора мутации используется оператор swap. Используется циклический кроссинговер [9], для которого выполняется свойство передачи генов в терминах наследования значений в позициях перестановки. В операторе селекции особи выбираются с помощью s-турнирной селекции (извлекается s случайных особей и из них выбирается лучшая по целевой функции).

Генетический алгоритм GA

1. Построить начальную популяцию П.
2. Пока не выполнен критерий остановки:
    1. Выбрать оператором селекции две родительских особи π_1 и π_2.
    2. Применить к π_1 и π_2 оператор мутации.
    3. Применить к π_1 и π_2 оператор скрещивания и построить потомка π'.
    4. Заменить худшую особь в популяции на π'.

3. Экспериментальное исследование

Алгоритмы реализованы на языке Python с использованием встроенных библиотек. Эксперименты проводились на компьютере с процессором Intel Core i5-8400 2.8 GHz. Тесты проводились на серии задач с разной размерностью и плотностью графов, сгенерированных случайным образом на основе метода Эрдёша–Реньи. Размерности графов $G_1 = (V_1, E_1)$ и $G_2 = (V_2, E_2) : |V_1| = 100$ и $|V_2| = 30$, $|V_1| = 200$ и $|V_2| = 60$, $|V_1| = 100$ и $|V_2| = 50$, $|V_1| = 200$ и $|V_2| = 100$ с плотностью 0.3, 0.5 и 0.8 для каждой из задач. Параметры генетического алгоритма: размер популяции – 50, размер турнира – 5. Для статистического анализа результатов на каждой задаче алгоритмы запускались 30 раз на 20 тысяч итераций.

Результаты экспериментов в терминах значений целевой функции $f(\cdot)$ представлены в таблицах 1, 2, 3, где в эволюционной стратегии используется только один оператор мутации, а в таблице 4 представлен результат работы эволюционной стратегии с адаптивной схемой выбора операторов. В обозначении задачи указывается число вершин в большом графе, малом графе и плотность графов.

В таблице 5 представлены средние отклонения от верхней границы (количество рёбер в малом графе) для всех исследуемых версий эволюционной стратегии и генетического алгоритма. Для оценки различий в целевой функции между значениями, найденными двумя $(1+1)-ES$: с оператором swap и адаптивной схемой, использовался тест Вилкоксона. Лучшие найденные значения для каждой задачи выделены жирным шрифтом, а символ «*» указывает на статистическую значимость отличий при уровне значимости $\alpha = 0.05$. Генетический алгоритм демонстрирует результаты сопоставимые с адаптивной эволюционной стратегией (результаты в среднем не более чем на 2 % хуже).

Таблица 1. Результаты $(1+1)$-ES с оператором swap

Задача	**мин.**	**макс.**	**средн.**
100/30_0.3	93	109	102,27
100/30_0.5	181	194	188,2
100/30_0.8	340	346	342,67
200/60_0.3	341	363	352,57
200/60_0.5	680	704	691,03
200/60_0.8	1329	1353	1340,83
100/50_0.3	230	252	243,03
100/50_0.5	465	485	473,8
100/50_0.8	911	933	923,13
200/100_0.3	824	859	840,73
200/100_0.5	1705	1760	1730,87
200/100_0.8	3566	3597	3581,83

Таблица 2. Результаты $(1+1)$-ES с оператором shift

Задача	**мин.**	**макс.**	**средн.**
100/30_0.3	73	87	81,4
100/30_0.5	157	172	164,07
100/30_0.8	321	333	327,4
200/60_0.3	254	289	274,4
200/60_0.5	577	609	593,3
200/60_0.8	1253	1288	1271,4
100/50_0.3	175	197	189,13
100/50_0.5	397	432	409,83
100/50_0.8	862	886	874,5
200/100_0.3	636	684	657,03
200/100_0.5	1487	1554	1512,5
200/100_0.8	3394	3446	3418,6

Таблица 3. Результаты $(1 + 1)$-ES с оператором shuffle

Задача	**мин.**	**макс.**	**средн.**
100/30_0.3	85	94	89,2
100/30_0.5	166	183	175,4
100/30_0.8	328	337	333,6
200/60_0.3	288	305	296,9
200/60_0.5	604	641	624,17
200/60_0.8	1284	1311	1296,83
100/50_0.3	196	217	205,83
100/50_0.5	422	442	432,67
100/50_0.8	884	902	893
200/100_0.3	687	726	703,73
200/100_0.5	1551	1592	1571,6
200/100_0.8	3444	3480	3465,3

Таблица 4. Результаты адаптивной версии $(1 + 1)$-ES

Задача	**мин.**	**макс.**	**средн.**
100/30_0.3	96	107	101,4
100/30_0.5	186	195	190,7
100/30_0.8	336	347	342,7
200/60_0.3	339	359	348,6
200/60_0.5	671	699	684,4
200/60_0.8	1327	1350	1340
100/50_0.3	227	254	242,9
100/50_0.5	464	484	472
100/50_0.8	910	933	924
200/100_0.3	810	847	830,7
200/100_0.5	1698	1735	1719
200/100_0.8	3562	3589	3575

Таблица 5. Отклонение среднего значения целевой функции от верхней границы (число рёбер в малом графе)

Задача	Верхняя граница	(1+1)-ES swap	(1+1)-ES shift	(1+1)-ES shuffle	(1+1)-ES adaptive	GA
100/30_0.3	130	21,33%	37,38%	31,38%	22,02%	**21,28%**
100/30_0.5	217	13,27%	24,39%	19,17%	**12,12%***	13,21%
100/30_0.8	348	1,53%	5,92%	4,14%	**1,52%**	1,43%
200/60_0.3	531	**33,60%***	48,32%	44,09%	34,36%	36,02%
200/60_0.5	885	**21,92%***	32,96%	29,47%	22,67%	23,9%
200/60_0.8	1416	**5,31%**	10,21%	8,42%	5,40%	5,88%
100/50_0.3	367	**33,78%**	48,47%	43,92%	33,81%	33,99%
100/50_0.5	612	**22,58%**	33,03%	29,30%	22,88%	23,9%
100/50_0.8	980	5,80%	10,77%	8,88%	**5,71%**	5,71%
200/100_0.3	1485	**43,39%***	55,76%	52,61%	44,06%	45,17%
200/100_0.5	2475	**30,07%***	38,89%	36,50%	30,53%	31,49%
200/100_0.8	3960	**9,55%***	13,67%	12,49%	9,73%	10,13%

По результатам, представленным в таблицах выше, можно сделать вывод, что оператор swap оказался эффективнее других операторов и показал схожие результаты с адаптивной схемой выбора операторов, но адаптивная схема имеет преимущество, так как может использовать несколько операторов, эффективность которых заранее неизвестна.

По сериям примеров разной размерности можно сделать следующие выводы:

1) на графах с плотностью 0.8 алгоритмы показывают более стабильные результаты с точки зрения целевой функции и это ожидаемо, так как, чем больше связей между вершинами в графе, тем больше шансов найти подграф, соответствующий малому графу;

2) на графах с плотностью 0.3 алгоритмы ведут себя менее стабильно, общее качество снижается;

3) при работе с более плотным графом алгоритмы демонстрируют меньший разброс значений и более высокие показатели качества решений.

Заключение

Проведено исследование оптимизационной версии задачи граф-подграф изоморфизма. Проанализированы вычислительная сложность и существующие методы, на основе чего разработана новая эволюционная стратегия с использованием набора операторов мутации и генетический алгоритм для указанной задачи. Реализована адаптивная схема выбора операторов, учитывающая их эффективность в процессе работы эволюционной стратегии. В результате экспериментального исследования и статистического анализа результатов показана применимость предложенных ал-

горитмов. Дальнейшие исследования могут быть направлены на исследование взвешенного варианта задачи и учёта дополнительных ограничений при сопоставлении вершин.

Благодарности

Работа выполнена при поддержке Математического Центра в Академгородке, соглашение с Министерством науки и высшего образования Российской Федерации № 075-15-2025-349 от 29.04.2025.

Литература

1. Демченко М.В., Каширина И.Л., Фирюлина М.А. Использование методов обучения с подкреплением в задачах медицинской практики // Вестник ВГУ, серия: системный анализ и информационные технологии. 2022. № 1. С. 111–117.
2. Дидешко Д.С., Кравцов Г.Р., Захарова Ю.В. Об одном подходе к решению задачи граф-подграф изоморфизма // Молодёжь третьего тысячелетия: сборник научных статей XLIX региональной студенческой научно-практической конференции: в 3 ч. 2025. С. 224-227.
3. Еремеев А.В. Конспект лекций по курсу «Эволюционные алгоритмы». Омск, 2024. 75 стр.
4. Ильяшенко М.Б. Алгоритм нахождения граф-подграф изоморфизма для взвешенных графов и его применение // Радиоэлектроника. Информатика. Управление. № 1, 2007. С. 62–68.
5. Cho M., Lee J., Lee K. M. Reweighted random walks for graph matching // European conference on Computer vision. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. P. 492–505.
6. Haque M.N., Mathieson L., Moscato P. A memetic algorithm approach to network alignment: mapping the classification of mental disorders of DSM-IV with ICD-10 // Proceedings of the Genetic and Evolutionary Computation Conference. 2019. P. 258–265.
7. Holland J.H. Adaptation in Natural and Artificial Systems // An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT Press, 1992. P. 89–118.
8. Klau G. W. A new graph-based method for pairwise global network alignment // BMC bioinformatics. 2009. V. 10. P. 1–9.
9. Reeves C. R. Genetic algorithms for the operations researcher // INFORMS Journal of Computing. 1997. V. 9, No. 3. P. 231–250.

EVOLUTIONARY ALGORITHMS FOR THE OPTIMIZATION VARIANT OF THE GRAPH-SUBGRAPH ISOMORPHISM PROBLEM

Yu.V. Zakharova[1, 2]
Ph.D. (Phys.-Math.), Senior Researcher, e-mail: yzakharova@ofim.oscsbras.ru
V.P. Lobanov[1, 2]
Ph.D. Student, e-mail: lobanov.vasiliy.pavlovich@gmail.com
G.R. Kravtsov[3]
Student, e-mail: grk55rus@mail.ru

[1]Sobolev Institute of Mathematics, Omsk Department, Omsk, Russia
[2]Novosibirsk State University, Novosibirsk, Russia
[3]Omsk State Technical University, Omsk, Russia

Abstract. The article proposes evolutionary algorithms for the optimization version of the graph-subgraph isomorphism problem, which has applications in computer systems and bioinformatics. The genetic algorithm uses operators based on inheriting or mutating values at positions within permutations used to encode solutions. The evolutionary strategy employs an adaptive selection of operators. Experimental studies demonstrate the applicability of the developed algorithms.

Keywords: evolutionary strategy, genetic algorithm, isomorphism, experiment.

Дата поступления в редакцию: 26.01.2026