УДК 004.021

АЛГОРИТМ НАХОЖДЕНИЯ НАИБОЛЬШЕГО ОБЩЕГО ПОДГРАФА

М.Б. ИЛЬЯШЕНКО

Предлагается новый переборный алгоритм решения задачи нахождения наибольшего общего подграфа. Приведены результаты численного анализа производительности алгоритма на графах различных классов и размеров, входящих в состав базы графов для оценки производительности алгоритмов решения задач установления морфизма на графах. Дана оценка потенциала применения разработанного алгоритма для решения реальных прикладных задач на графах размером порядка сотен вершин

Графы широко применяются в различных областях науки и техники: в системах распознавания образов, химии, компьютерных системах и сетях, логистике и т.д. Если графы применяются для представления структурных объектов, то степень сходства объектов становится равнозначна определению сходства графов, их представляющих. Изоморфизм графов используется для определения идентичности структур двух графов [1]. Более общая операция граф–подграф изоморфизма используется для определения, является ли один граф структурной частью другого [1,2]. В работах [3,4] рассмотрены вопросы применения понятия наибольшего общего подграфа для определения степени схожести графов.

Нахождение наибольшего общего подграфа (НОП) двух заданных графов — задача известная. В работах [5, 6] алгоритм нахождения НОП применялся для сравнения молекул. В работе [7] предложен алгоритм нахождения НОП, использующий метод поиска с возвратом. Другим, кардинально отличным подходом к нахождению НОП, является поиск максимальной клики в модульном произведении графов [8, 9].

Проблемы нахождения НОП и максимальной клики являются NP-полными задачами [10]. Поэтому было предложено множество эвристических алгоритмов решения. Обзор таких алгоритмов, в том числе анализ их вычислительной сложности и потенциальных возможностей применения, сделан в работе [11]. Например, в [12] приводятся результаты численного исследования двух алгоритмов нахождения НОП на основе базы случайных графов. В частности, произведено сравнение производительности переборного алгоритма и алгоритма на основе нахождения максимальной клики в модульном произведении графов, показавшее, что при низкой плотности реберного заполнения графов более эффективным является подход, основанный на переборном алгоритме.

В данной статье приводится описание алгоритма нахождения НОП, построенного на основе метода ветвей и границ. Приводится детальное описание алгоритма, а также результаты исследования его производительности на графах эталонной базы для сравнения алгоритмов установления изоморфности.

АЛГОРИТМ

Максимальный общий подграф. Пусть даны графы $G_1 = (V_1, E_1)$ и $G_2 = (V_2, E_2)$. Графы $G_1' \subseteq G_1$ и $G_2' \subseteq G_2$. При этом подграф G_1' изоморфен подграфу G_2'. G_1' и G_2' содержат максимальное число вершин среди всех возможных подграфов.

Алгоритм поиска НОП удобно описывать в терминах поиска в пространстве состояний. Каждое состояние s процесса совмещения вершин соответствует частичной подстановке $\varphi(s)$, которая содержит лишь часть вершин полной подстановки. Каждому состоянию соответствуют подграфы $G_1(s)$ и $G_2(s)$, полученные из вершин графов G_1 и G_2, вошедших в частичную подстановку $\varphi(s)$, и ребер, соединяющих эти вершины. В дальнейшем обозначим $\varphi_1(s)$ и $\varphi_2(s)$ проекции подстановки $\varphi(s)$ на V_1 и V_2.

Алгоритм состоит из предварительной и основной частей.

В предварительной части алгоритма вершины графа G_2 сортируются в порядке убывания степени связности вершин. Первой ставится вершина с наибольшим числом инцидентных ребер. Среди оставшихся вершин каждый раз выбирается та, которая имеет максимальное число ребер, соединяющих ее с уже совмещенными вершинами. Таким образом, вначале идут вершины, имеющие большее число внутренних связей. Этот порядок следования вершин делает более жесткими граничные условия, используемые в переборной части алгоритма и, в конечном счете, сужает область поиска.

Центральной является переборная часть алгоритма, основанная на рекурсивной функции последовательного наложения вершин с возвратом. Вершины графа G_2 остаются нетронутыми, и им в соответствие ставятся вершины графа G_1.

В алгоритме используются следующие векторы индексов вершин графов, инцидентных совмещенным вершинам:

In_1 — входящие графа G_1; Out_1 — исходящие G_1; In_2 — входящие G_2; Out_2 — исходящие G_2.

Пусть на данном шаге в подстановку предполагается включить вершины V_1^i и V_2^i. Коэффициент, используемый при изменении индексов входящих и исходящих вершин, определяется как $k = 2^{\mathrm{mod}(1,31)}$. Тогда

для $\forall(V_1^j, V_1^i) \in E_1$, $\forall(V_1^i, V_1^j) \in E_1$, $\forall(V_2^j, V_2^i) \in E_2$, $\forall(V_2^i, V_2^j) \in E_2$ изменяем значения соответственно $In_1^j = In_1^j + k$, $Out_1^j = Out_1^j + k$, $In_2^j = In_2^j + k$, $Out_2^j = Out_2^j + k$.

Такое изменение индексов позволяет хранить в векторах In_1, In_2, Out_1 и Out_2 не только количество связей каждой вершины с вершинами, уже вошедшими в подстановку, но и косвенно, через индекс k — номер итерации, на которой данная вершина была добавлена. Использование k в пределах от 20 до 231 обусловлено применяемым в языке программирования C++

32-разрядным типом данных. С введением 64-разрядных типов данных, поддерживаемых на аппаратном уровне, разброс значений коэффициента k можно будет увеличить.

На каждом шаге переборной части алгоритма среди вершин графа G_1, еще не вошедших в частичную подстановку $\varphi(s)$, выбирается вершина V_1^i, потенциально совместимая с очередной вершиной V_2^j графа G_2. Перед совмещением проверяются условия допустимости добавления пары вершин $\left(V_1^i, V_2^j\right)$ в частичную подстановку.

- $In_1^i = In_2^j$.
- $Out_1^i = Out_2^j$.
- Множества ребер $E_1' = \left(V_1^i, \varphi_1(s)\right)$ и $E_2' = \left(V_2^j, \varphi_2(s)\right)$ должны совпадать, т.е. $E_1' = E_2'$.

Последнее условие является прямым следствием постановки задачи поиска НОП. Если очередная полученная подстановка $\varphi(s)$ больше, чем текущая максимальная частичная подстановка $\varphi'(s)$, то $\varphi'(s) = \varphi(s)$.

Если на очередном шаге не удается найти совместимых вершин, то выполняется операция отката (backtracking), восстанавливаются значения векторов In_1, In_2, Out_1 и Out_2 предыдущего шага, и поиск продолжается, начиная со следующей пары еще не совмещенных вершин.

БАЗА ГРАФОВ ДЛЯ ОЦЕНКИ ПРОИЗВОДИТЕЛЬНОСТИ АЛГОРИТМОВ НАХОЖДЕНИЯ НОП

Для сравнения производительности алгоритмов проверки на изоморфность, установления граф–подграф изоморфизма и нахождения НОП предложена эталонная база графов [13] (см. таблицу).

Наборы для различных размеров подграфов (10, 30, 50, 70 и 90% от большего) одинаковы. Всего протестировано 50000 пар графов.

Типы графов, вошедшие в состав эталонной базы, учитывают особенности реальных задач, которые решаются с применением алгоритма нахождения НОП. Так для задач распознавания образов, распределения нагрузки в компьютерных сетях, синтеза печатных плат и анализа топологии более характерны графы с небольшой степенью вершин, низкой плотностью реберного заполнения. Такие графы представлены в виде наборов графов с ограниченной степенью вершин и нерегулярных сетей. Регулярные (сильнорегулярные) графы, как правило, являются наихудшим случаем для переборных алгоритмов решения задач морфизма на графах и характеризуют уровень производительности алгоритма в «наихудшем случае». Такие графы представлены наборами регулярных графов с ограниченной степенью вершин и регулярных сетей. Кроме того, представлены случайные графы для оценки производительности алгоритма на графах, не обладающих специальными свойствами.

Состав набора графов эталонной базы

Тип графов	**Количество пар**	**Параметры**	**Размеры (вершин)**
Случайно сгенерированные графы	1100	$n = 0{,}005$	20–100
	1100	$n = 0{,}01$	20–100
	1100	$n = 0{,}02$	20–100
Регулярные сети	700	2D сеть	40–100
	500	3D сеть	60–100
Нерегулярные сети	800	Нерегулярные 2D сети ρ={0,2; 0,4; 0,6}	40–100
	1500	Нерегулярные 3D сети ρ={0,2; 0,4; 0,6}	60–100
Регулярные графы с ограниченной степенью вершин	700	Степень=3	40–100
	400	Степень=6	70–100
Нерегулярные графы с ограниченной степенью вершин	700	Степень=3, α=0,1	40–100
	400	Степень=6, α=0,1	70–100
Итого	10000		

ИССЛЕДОВАНИЕ ПРОИЗВОДИТЕЛЬНОСТИ АЛГОРИТМА НА ГРАФАХ ЭТАЛОННОЙ БАЗЫ

Исследования производительности алгоритма осуществлялись на компьютере AMD Athlon 1700+, 512MB RAM. Компилятор Visual C++ 6.0.

Различные наборы графов обозначим следующим образом:

MCS_0 — набор графов с размером подграфа 10%, MCS_1 — 30%, MCS_2 —50%, MCS_3 —70%, MCS_4 —90%.

Характерные результаты исследования приведены на рис. 1–10. Для удобства сравнения производительности различных алгоритмов приведены как среднее время сравнения одной пары графов, так и среднее число перебранных узлов дерева решений рекурсивного алгоритма. При сравнении переборных алгоритмов число перебранных узлов дерева решений выступает более адекватным критерием сравнения, так как время вычислений может сильно зависеть от особенностей программной реализации и производительности компьютера, на котором выполнялись исследования.

Результаты исследования показали, что для всех графов эталонной базы алгоритм находит решение за время, не превышающее 0,2 с, перебрав не более 6000 узлов дерева решений. Такая производительность дает возможность использовать разработанный алгоритм нахождения НОП в реальных прикладных задачах. Характерной особенностью алгоритма является повышение производительности с ростом размера меньшего графа. Это связано с появлением дополнительной информации в переборной части алгоритма за счет увеличения числа ребер в графах, что обусловливает больший разброс значений векторов In_1, In_2, Out_1, Out_2 и как следствие — сужение области поиска за счет усиления граничного условия переборной части алгоритма.

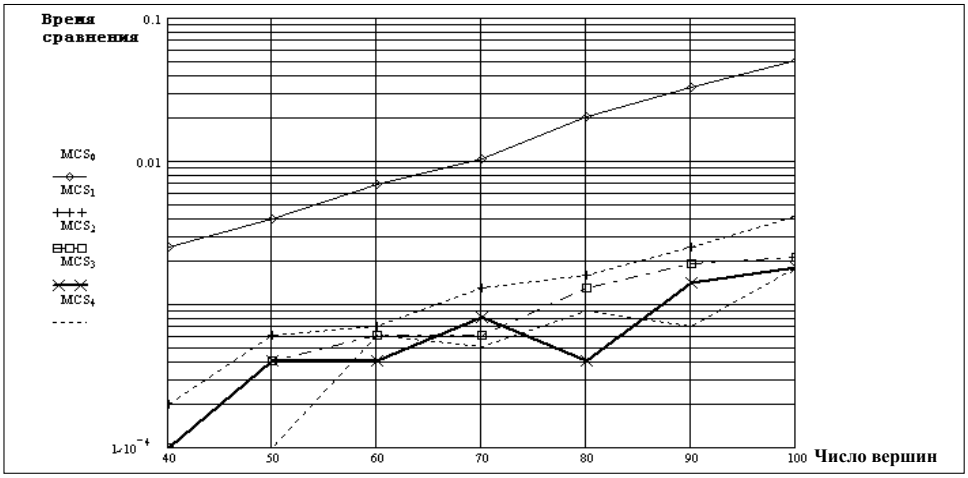

Рис. 1. Время проверки регулярных графов с ограниченной степенью вершин (*N*=40...100, степень=3)

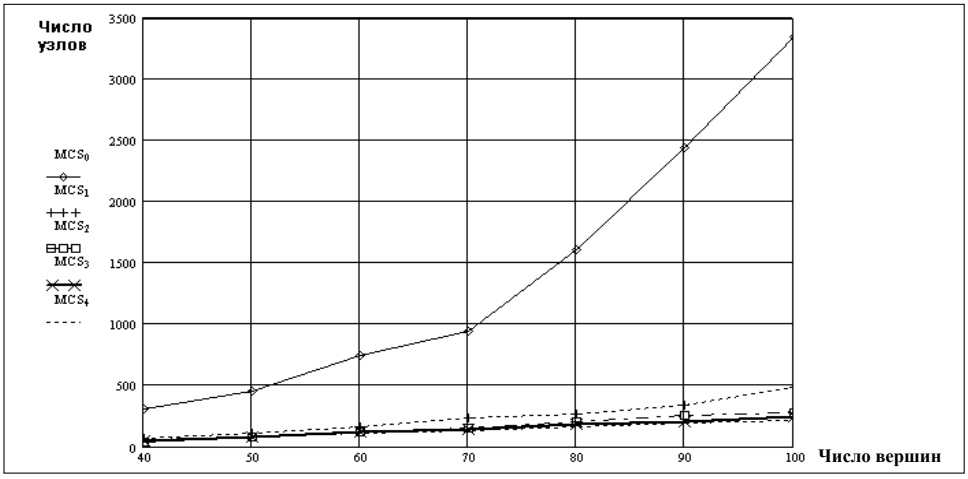

Рис. 2. Число перебранных узлов при проверке регулярных графов с ограниченной степенью вершин (*N*=40...100, степень=3)

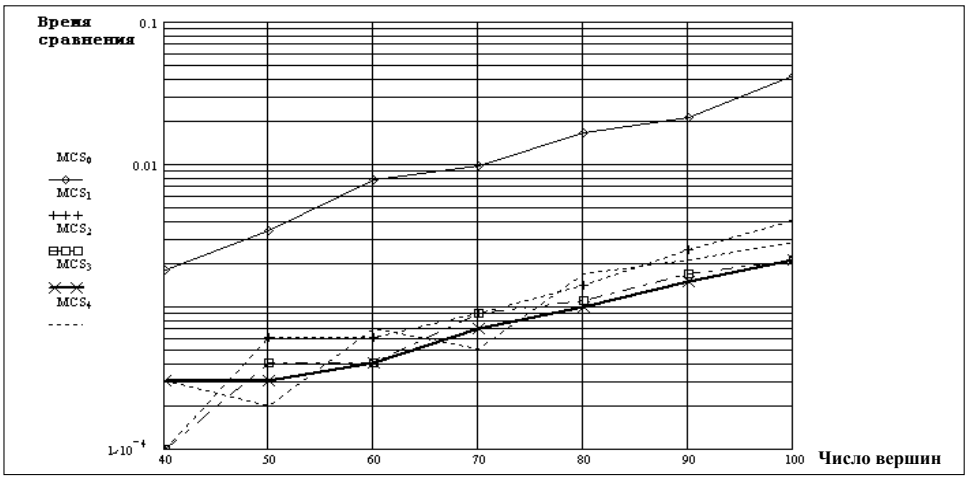

Рис. 3. Время проверки нерегулярных графов с ограниченной степенью вершин (*N*=40…100, степень=3, *a*=0,1)

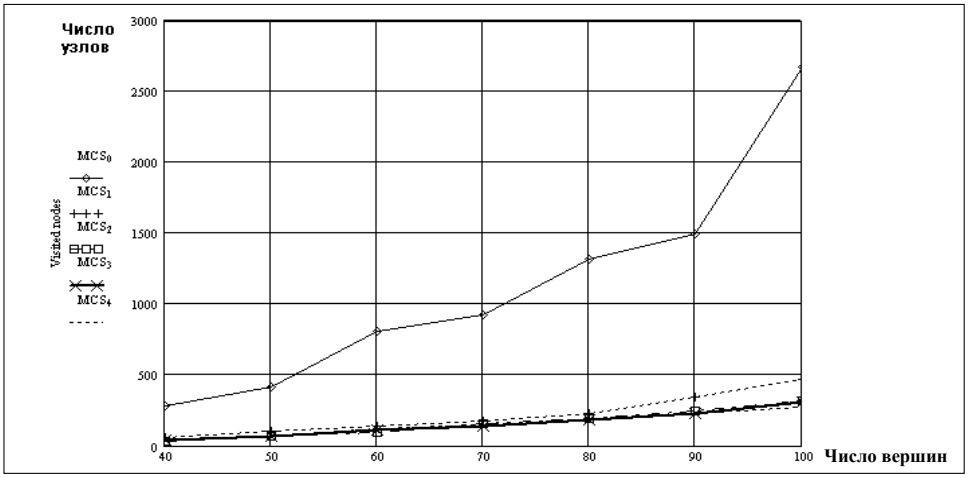

Рис. 4. Число перебранных узлов при проверке нерегулярных графов с ограниченной степенью вершин (N=40…100, степень=3, a=0,1)

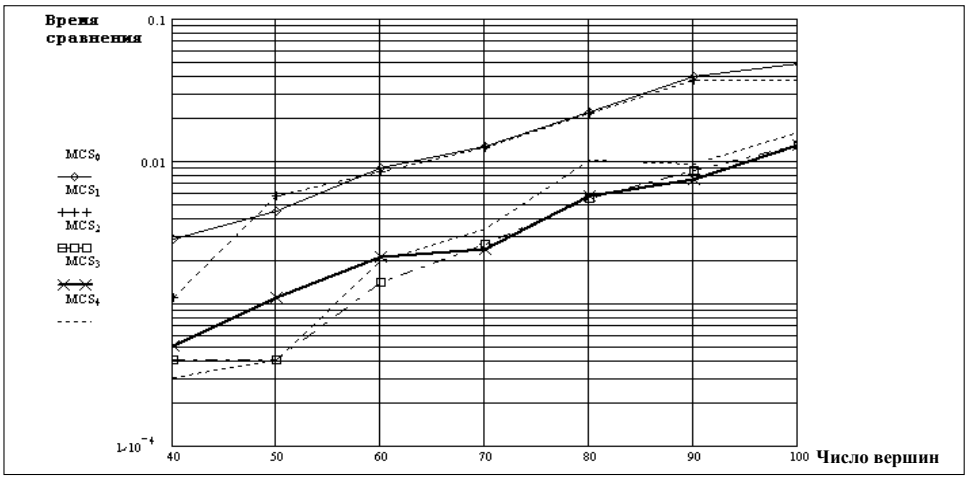

Рис. 5. Время проверки регулярных 2D сетей (N=40…100, p=0,0)

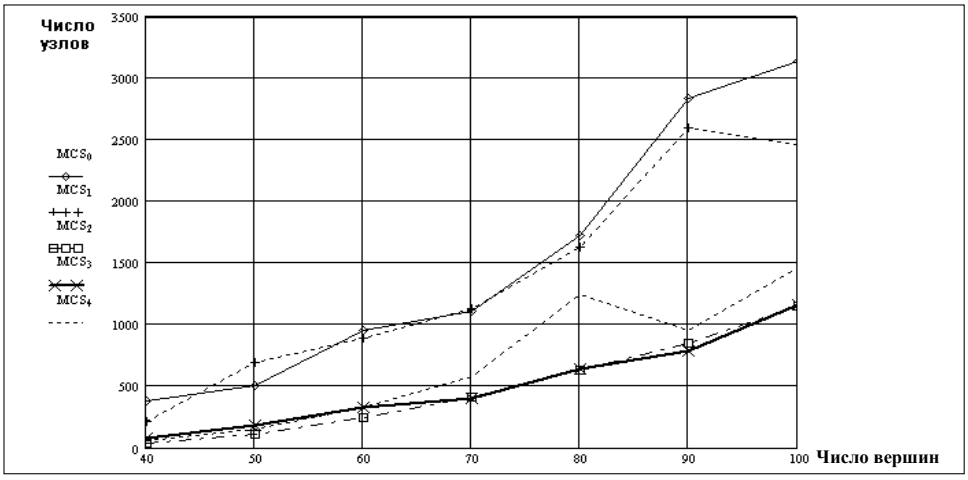

Рис. 6. Число перебранных узлов при проверке регулярных 2D сетей (N=40…100, p=0,0)

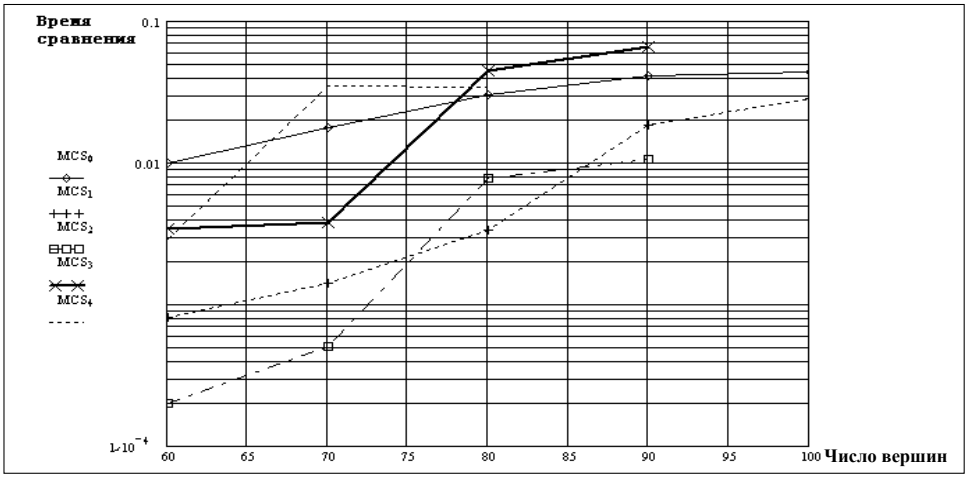

Рис. 7. Время проверки регулярных 3*D* сетей (*N*=60…100, *p*=0,0)

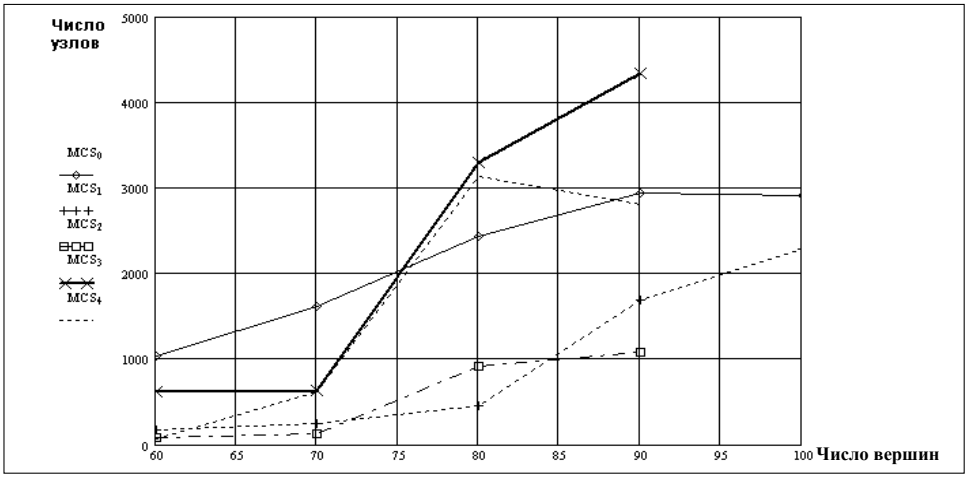

Рис. 8. Число перебранных узлов при проверке регулярных 3*D* сетей (*N*=60…100, *p*=0,0)

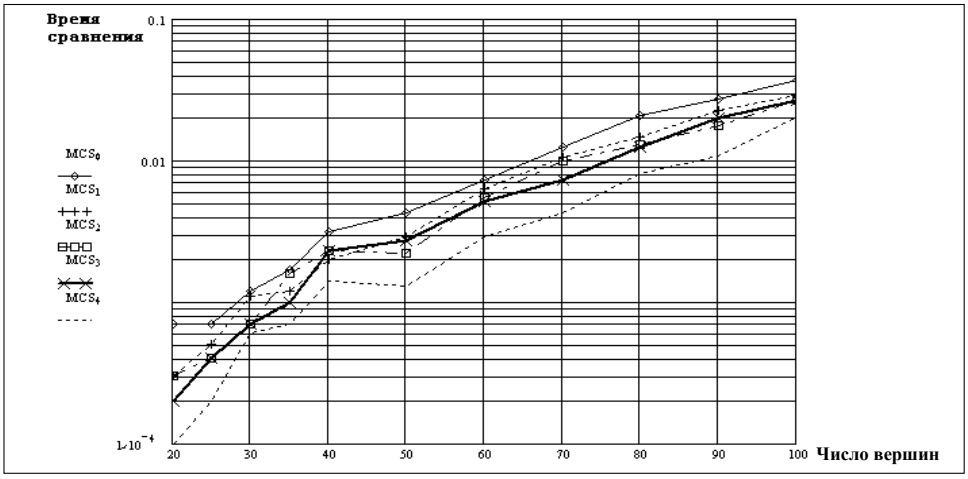

Рис. 9. Время проверки случайно сгенерированных графов (*N*=20…100, *n*=0,01)

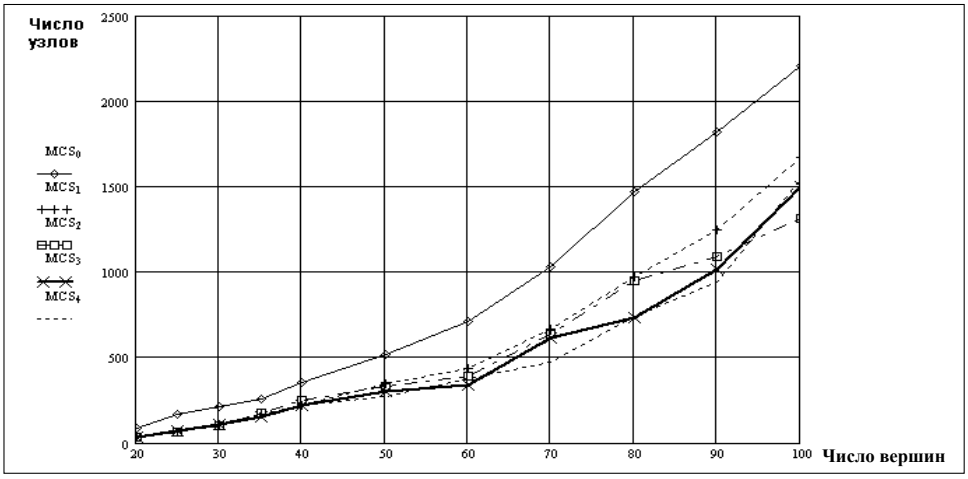

Рис. 10. Число перебранных узлов при проверке случайно сгенерированных графов (N=20…100, n=0,01)

ВЫВОДЫ

Представлен новый алгоритм нахождения наибольшего общего подграфа. Приведены результаты численного анализа производительности алгоритма на эталонной базе графов для оценки производительности алгоритмов морфизма на графах. Результаты исследования показали, что разработанный алгоритм в состоянии сравнивать графы размерностью до 100 вершин за время, не превышающее 0,2 с, при переборе не более 6000 узлов дерева решений. Графы большего размера в эталонной базе на данный момент отсутствуют.

Производительность разработанного алгоритма позволяет применять его для решения реальных прикладных задача на графах размерностью порядка сотен вершин.

Дальнейшие усилия будут направлены на численное определение вычислительной сложности алгоритма путем исследования его производительности на графах большего размера, чем представленные в эталонной базе, а также на более детальное сравнение с аналогами и расширение функциональности для решения задач на взвешенных графах и связанными с ними проблемами оптимизации.

ЛИТЕРАТУРА

1. *Ullmann J.R.* An Algorithm for Subgraph Isomorphism // Journal of the Association for Computing Machinery. — 1976. — № 23. — P. 31–42.
2. *An improved* algorithm for matching large graphs / L.P. Cordella et al. // Proceedings of the 3rd IAPR-TC-15 International Workshop on Graph-based Representations. — Italy, 2001. — P. 149–159.
3. *On the* Minimum Supergraph of Two Graphs / H. Bunke, X. Jiang, A. Kandel // Computing. — 2000. — **65**. —P. 13–25.
4. *Bunke H., Sharer K.* A Graph Distance Metric Based on the Maximal Common Subgraph // Pattern Recognition Letters. — 1998. — **19**. — № 3, 4. — P. 255–259.

5. *Levi G.* A Note on the Derivation of Maximal Common Subgraphs of Two Directed or Undirected Graphs // Calcolo. — 1972. —**9**. — P. 341–354.
6. *Molecular* Structure Comparison Program for the Identification of Maximal Common Substructures / M. M. Cone, Rengachari Venkataraghven, F. W. McLafferty // Journal of Am. Chem. Soc. — 1977. — **99**, №23. — P. 7668–7671.
7. *McGregor J.J.* Backtrack Search Algorithms and the Maximal Common Subgraph Problem // Software Practice and Experience. — 1982. — **12**. — P. 23–34.
8. *Bron C., Kerbosch J.* Finding All the Cliques in an Undirected Graph // Communication of the Association for Computing Machinery. — 1973. — **16**. — P. 575–577.
9. *Immanuel M. Bomze*. Evolution towards the Maximum Clique // Journal of Global Optimization. — 1997. — **10**, № 2. — P. 143–164.
10. *Garey M.R., Johnson D.S.* Computers and intractability: a guide to the theory of NP-completeness. — S.F.: W.H. Freeman and Comp.,1979. — 251 p.
11. *The Maximum* Clique Problem / I.M. Bomze et al. // Handbook of Combinatorial Optimization. — Kluwer Academy Pub. — 1999. — **4**. — P. 656.
12. *A Comparison* of Algorithms for Maximum Common Subgraph on Randomly Connected Graphs / H. Bunke at al. // Structural, Syntactic, and Statistical Pattern Recognition. — Springer Berlin: Heidelberg. — 2002. — P. 123–132.
13. *A large database* of graphs and its use for benchmarking graph isomorphism algorithms / De Santo M. et al. // Pattern Recogn. — 2003 — **24,** № 8. — P. 1067–1079.

Поступила 17.04.2007