Дудко Сергей Анатольевич – Южный федеральный университет; e-mail: dudko@sfedu.ru; 347900, г. Таганрог, пер. Тургеневский, 44; тел.: +79034318173; Кафедра интеллектуальных и многопроцессорных систем; аспирант.

Dudko Sergei Anatolievich – Southern Federal University, e-mail: dudko@sfedu.ru; 44, Turgenevskii lane, Taganrog, 347900, Russia; phone: +79034318173; the department of intellectual and multiprocessor systems; graduate student.

УДК 004.382.2 DOI 10.18522/2311-3103-2020-7-121-129

А.В. Касаркин

МЕТОД РЕШЕНИЯ ГРАФОВЫХ NP-ПОЛНЫХ ЗАДАЧ НА РЕКОНФИГУРИРУЕМЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ НА ОСНОВЕ ПРИНЦИПА РАСПАРАЛЛЕЛИВАНИЯ ПО ИТЕРАЦИЯМ

При решении графовых NP-полных задач на многопроцессорных системах рост оборудования не приводит к пропорциональному росту производительности системы, поэтому не всегда удается решить задачу за приемлемое время. Целью работы, описанной в статье, является минимизация времени решения задачи поиска максимальных клик графа с использованием реконфигурируемых вычислительных систем (РВС). При решении задачи на РВС методом распараллеливания по слоям рост производительности также замедляется, несмотря на лучшую степень масштабируемости по сравнению с многопроцессорными реализациями. В статье предложен метод создания параллельно-конвейерных программ для реконфигурируемых вычислительных систем на основе распараллеливания по итерациям для решения графовых NP-полных задач. Рассмотрено, что использовать битовый способ представления множеств (как в методе распараллеливания по слоям) для метода распараллеливания по итерациям не является эффективным. Новый метод отличается организацией вычислений, а именно – обработкой неупорядоченных множеств, доступ к элементам которых осуществляется не по адресам (как в массивах), а по значениям (именам вершин и именам дуг графа). Показано, что новый метод на основе распараллеливания по итерациям, несмотря на более низкую удельную производительность, связанную с тем, что вычислительным подструктурам из-за символьного представления множеств необходимо обработать большее число промежуточных данных, обеспечивает практически линейный рост реальной производительности РВС при значительно большем количестве вычислительного ресурса по сравнению с методом распараллеливания по слоям.

Теория множеств; графовые NP-полные задачи; задача о клике; максимальные клики графа; реконфигурируемые вычислительные системы; программируемые логические интегральные схемы (ПЛИС); информационный граф; структурная реализация; конвейер; суперкомпьютеры.

A.V. Kasarkin

A METHOD FOR SOLVING GRAPH NP-COMPLETE TASKS ON RECONFIGURABLE COMPUTER SYSTEMS BASED ON THE ITERATION PARALLELIZING PRINCIPLE

When we solve graph NP-complete tasks on multiprocessor systems, the growth of hardware resource does not lead to the proportional increase of the system performance, and hence, the task solution time is not always reasonable. The aim of our research, given in the paper, is minimization of the solution time of the task of maximal clique enumeration on reconfigurable computer systems (RCS). When we solve tasks on RCSs with the help of the method of parallelizing by layers, the growth of performance also slows down in spite of better scalability in comparison with multiprocessor implementations. In the paper, we suggest a method of parallel-pipeline application development for reconfigurable computer systems. The method is based on parallelizing by

layers for graph NP-complete tasks. We show that the bit representation of sets, which is used for the method of parallelizing by layers, is not efficient for the method of parallelizing by iterations. The new method has another organization of calculations; it processes unordered sets, whose elements are accessed not by addresses (as in arrays), but by values (names of vertices and names of edges of the graph). We show that the new method, based on parallelizing by iterations, provides ramping of the RCS real performance at much larger computational resource in comparison with the method of parallelizing by layers. Its specific performance is lower, because computing substructures are to process more intermediate data due to symbolic representation of sets.

Theory of sets; graph NP-complete tasks; clique task; maximal cliques of a graph; reconfigurable computer system; FPGA; information graph; structural implementation; pipeline; supercomputer.

Введение. Теория графов позволяет решать актуальные на сегодняшний день задачи из таких областей, как анализ социальных и вычислительных сетей, биоинформатика, логистика и многих других [1, 2]. Особое место занимают графовые задачи класса NP-полные, которые зачастую приходится решать быстрыми, но приближенными алгоритмами из-за длительного времени точного решения. Однако приближенное решение, в отличие от точных методов, не всегда приводит к оптимальному результату. В настоящее время одним из основных точных методов решения NP-полных задач является метод ветвей и границ [3]. Отличие данного метода от полного перебора заключается в исключении из обработки подмножеств возможных решений, которые заведомо не приведут к оптимальному результату. Тем не менее вычислительная сложность алгоритмов, основанных на методе ветвей и границ для решения NP-полных задач, остается экспоненциальной.

Поэтому графовые NP-полные задачи (далее – GNPC-задачи) являются наиболее трудновычислимыми. Кроме того, эти задачи характеризуются превалированием количества информационных обменов над количеством вычислений и требуют таких режимов доступа к памяти, которые приводят к нелинейной адресации и большому временному интервалу между повторными чтениями одних и тех же ячеек памяти (плохая пространственно-временная локализация приложений).

При применении стандартных методов решения GNPC-задач на многопроцессорных системах возникает ряд проблем, связанных с их масштабируемостью [4], которые приводят к тому, что при увеличении числа процессоров рост производительности замедляется, а при достижении некоторого их числа существенно замедляется [5-10]. На рис. 1 в качестве примера приведены графики, иллюстрирующие ускорение решения при распараллеливании задачи поиска максимальных клик [11] в зависимости от числа используемых ядер процессора [9, 10].

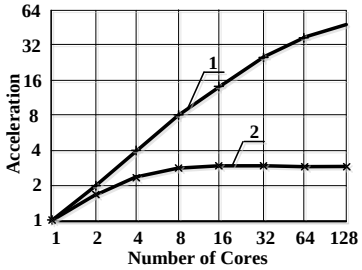

Рис. 1. Графики ускорения времени решения в зависимости от числа используемых ядер процессора: 1 – для графа с 300-ми вершинами из набора тестов «dimacs» (phat300-2) [12]; 2 – для графа Муна–Мозера [13] с 51-й вершиной

Замедление роста производительности связано с тем, что при превышении некоторого числа процессорных ядер затраты на организацию межпроцессорного обмена начинают превышать затраты на собственно вычисления, а транзакционная организация вычислений не позволяет начать выполнять новую операцию, пока процессор не завершит предыдущую операцию.

Для эффективного обмена данными между узлами обработки необходимо использовать вычислительную архитектуру, которая аппаратно поддерживает пространственную коммутацию. Такую возможность обеспечивают реконфигурируемые вычислительные системы (РВС) [14,15], при этом для решения на РВС задачу необходимо представить в виде информационного графа, определяющего вычислительную структуру параллельной программы и правила подачи данных на вход вычислительной структуры [16].

Важно отметить, что информационный граф GNPC-задач при решении методом ветвей и границ определяется не только алгоритмом обработки, но и промежуточными результатами вычислений: структура последующих слоев информационного графа определяется результатами вычислений на предыдущих слоях, то есть информационный граф является функционально нерегулярным. Поэтому для вычислительной структуры GNPC-задачи на РВС необходимо информационный граф привести к функционально-регулярному виду. Одним из способов приведения информационного графа к функционально регулярному виду является его дополнение до регулярной формы [16]. Однако граф, полученный таким способом, является функционально избыточным, что приводит к низкой удельной производительности. Более эффективным способом решения данной проблемы является процедура векторизации [16]. Применение данной процедуры в сочетании с модернизациями, направленными на перераспределение данных в процессе вычислений, как было показано в работах [17, 18], позволило разработать метод распараллеливания по слоям, который сократил время решения задачи поиска максимальных клик на РВС по сравнению с классической многопроцессорной системой. Однако с помощью программы имитационного моделирования вычислительных структур для решения на РВС графовых задач было установлено, что, несмотря на то что метод распараллеливания по слоям имеет в несколько раз лучшую степень масштабируемости по сравнению с методами для многопроцессорных систем, ему присущи те же проблемы – замедление роста производительности при превышении эффективного числа устройств (от 8 до 150 ПЛИС в зависимости от решаемого графа). В то же время существуют РВС [19], которые содержат в себе намного больший вычислительный ресурс, чем тот, при котором метод распараллеливания по слоям обеспечивает окололинейный рост производительности. Поэтому необходимо дальнейшее развитие методов синтеза вычислительной структуры для решения GNPC-задач на РВС.

Метод синтеза вычислительной структуры с распараллеливанием по итерациям для решения GNPC-задач на реконфигурируемых вычислительных системах. Еще одним эффективным методом синтеза вычислительной структуры является метод, в котором используется распараллеливание по итерациям [20]. При этом данные потоком проходят через вычислительную структуру.

Для приведения информационного графа GNPC-задачи к функционально-регулярному виду, позволяющему обеспечить синтез вычислительной структуры с распараллеливанием по итерациям, необходимо выделить базовый подграф [16], а затем информационный граф разделить на слои. При этом каждый слой содержит некоторое количество целых базовых подграфов. Затем в каждом слое все базовые подграфы заменяются одной вершиной. На рис. 2 слои разделены пунктирными эллипсами, а вершины $P_1 \ldots P_{31}$ являются базовыми подграфами.

В результате описанных действий информационный граф преобразуется в функционально-регулярную структуру, обеспечивающую распараллеливание по итерациям, в которой вершины последовательно соединены друг с другом информационными каналами.

На следующем шаге необходимо определиться с форматом обрабатываемых данных. При позиционном способе кодирования, в частности, при битовом способе представления множеств, операции, выполняемые над множествами, являются транзакционными, что приводит к простоям конвейера.

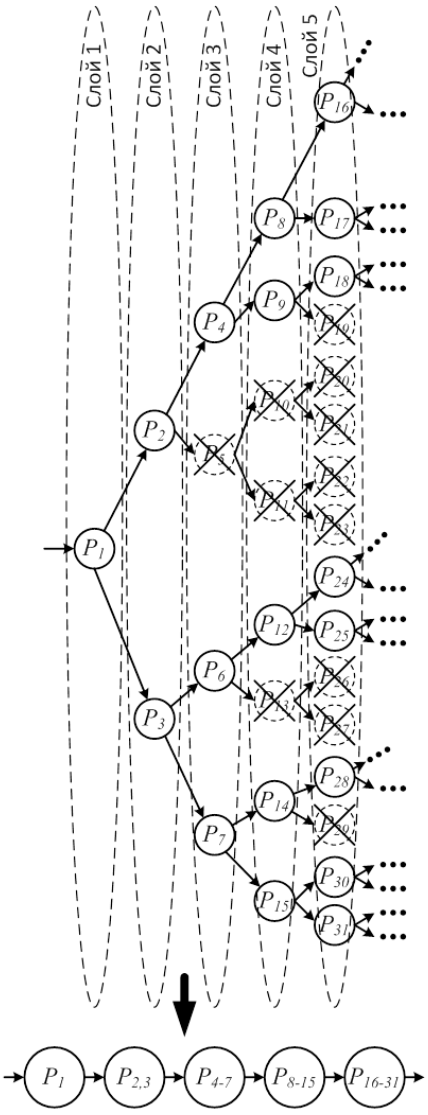

Рис. 2. Информационный граф GNPC-задач с группировкой вершин по порядку выполнения итераций

Поясним это на примере выполнения операции копирования, которая присутствует в алгоритме задачи поиска максимальных клик графа: скопировать любой элемент из множества K в переменную v. В процессе выполнения алгоритма

из-за удаления на каждой итерации элементов из множества K возникает ситуация, когда из множества K в переменную v необходимо скопировать последний оставшийся элемент множества, то есть K={k5}.

В битовом представлении позиция бита соответствует «номеру» вершины решаемого графа, а его значение является признаком присутствия либо отсутствия вершины во множестве. При битовом представлении множество можно сравнить со структурой данных «битовый массив», в которой каждая ячейка имеет свой определенный адрес, и этот адрес и является номером элемента множества. Для выполнения операции копирования элемента множества *K* в переменную *v* необходимо последовательно проверять элементы, начиная с первого элемента, и до нахождения элемента, который присутствует во множестве, таким образом, при битовой реализации результат можно получить только после проверки последнего элемента: k5. Это значит, что при битовом представлении множеств описанная операция является транзакционной: невозможно начать выполнять следующие операции, пока не будет проверен целиком весь массив – все множество K.

При символьном представлении множеств сам элемент множества является значением, элементы множества не упорядочены, и доступ к таким элементам осуществляется не по адресу, а по значению. Также при символьном представлении множество содержит имена только присутствующих в нем элементов, значит, для выполнения операции копирования элемента множества *K* в переменную *v* достаточно дождаться первого элемента множества *K* из потока и скопировать его в переменную *v*. Таким образом, для метода распараллеливания по итерациям битовое представление множеств вершин неэффективно и проигрывает символьному представлению, в котором каждому элементу множества соответствует некоторый символ или натуральное число.

При представлении множеств в символьном виде не представляется возможным провести векторизацию информационного графа для объединения базовых подграфов по слоям, так как генератор адресов не сможет адресоваться к конкретному элементу множества. Это связано с тем, что элементы во множествах располагаются в произвольном порядке, поэтому генератор адресов, который определяет порядок обработки промежуточных данных после проведения процедуры векторизации, не сможет вызвать нужный элемент, не перебрав множество полностью. Другой проблемой является то, что множества имеют переменную длину: длина меняется после прохождения каждой итерации, то есть генератору адресов нужно было бы решать проблему отделения одного множества в потоке от другого, не допускать перемешивания разных множеств при отправке их в конвейер.

Для решения указанной проблемы данные необходимо сгруппировать по множествам, чтобы в дальнейшем генератор адресов мог адресоваться целиком к вызываемому множеству. Таким образом, формируются потоки, состоящие из неупорядоченных множеств различной длины.

Полученная таким образом вычислительная структура представлена на рис. 3. Здесь $P_1 \ldots P_S$ – ступени вычислительного конвейера, каждая из которых соответствует одной итерации алгоритма; $y_1 \ldots y_S$ и $x_1 \ldots x_S$ – каналы, по которым последовательно передаются элементы множеств, а сами множества – параллельно (например, набор множеств K, M и P для задачи поиска максимальных клик графа). Блоки BUF представляют собой память, которая предназначена для хранения промежуточных множеств – результатов обработки итерации. При возникновении запроса BUF передает накопленное множество целиком. Вызов нужного элемента из прочитанного из BUF множества осуществляется в вычислительных конвейерах P по значениям – именам элементов множества.

Обработку одного элемента множества можно сравнить с работой ассоциативной памяти: поиск вершины графа осуществляется не по адресу, а по имени. Однако, в отличие от ассоциативной памяти, такой поиск реализуется не во времени, а в пространстве за счет непрерывного потока имён вершин через вычислительные блоки. В результате этого по структуре движется поток имен – элементов множеств, при этом множества имеют переменную длину.

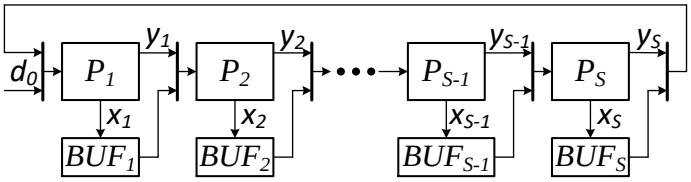

Рис. 3. Вычислительная структура, обеспечивающая распараллеливание по итерациям GNPC-задач

Поясним работу вычислительной структуры, представленной на рис. 5. В начале обработки на вход d_0 вычислительной структуры последовательно поступает начальный набор множеств. В процессе обработки через некоторое время, соответствующее латентности, конвейер P_1 начинает генерировать два новых набора множеств: первый набор - по шине y_1 последовательно, элемент за элементом, поступает на вход конвейера P_2, а второй - по шине x_1 последовательно запоминается в буфере BUF_1. После того как набор из P_1 полностью поступил в блок итерации P_2, на вход данной итерации подается набор множеств, который ранее запомнился в буфере BUF_1. Таким образом, на каждой итерации один из генерируемых наборов множеств идет на следующую итерацию, а второй попадает в буфер, где ожидает своей очереди для отправки на вход следующей итерации.

Оценка эффективности метода синтеза вычислительной структуры с распараллеливанием по итерациям для решения GNPC-задач на реконфигурируемых вычислительных системах. Для проверки работоспособности и эффективности предложенного метода была создана имитационная модель задачи поиска максимальных клик графа, моделирующая выполнение алгоритма таким образом, как он будет исполняться на РВС. Графики, отображающие результаты моделирования при решении графа на 500 вершин с вероятностью наличия ребра 0,6 на ПЛИС XCKU095, представлены на рис. 4 [21].

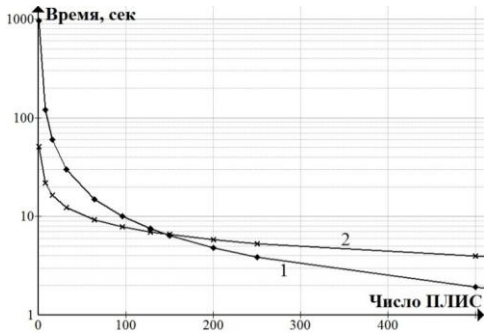

Рис. 4. График времени решения при распараллеливании задачи поиска максимальных клик: 1 – при использовании метода распараллеливания по слоям; 2 – при использовании метода распараллеливания по итерациям

На графиках заметно, что при превышении 145, ПЛИС программа, созданная с помощью метода распараллеливания по итерациям, решит задачу поиска максимальных клик в случайном графе на 500 вершин с вероятностью наличия ребра 0,6 быстрее, чем программа, созданная с помощью метода распараллеливания по слоям. Удельная производительность метода распараллеливания по итерациям при небольшом числе ПЛИС ниже из-за того, что при представлении множеств в символьном виде требуется обработать большее число промежуточных данных. В то же время ускорение метода распараллеливания по итерациям при увеличении вычислительного ресурса близко к линейному за счет значительного уменьшения коммутационных связей, что и обуславливает выигрыш метода при использовании более чем 145 ПЛИС.

Заключение. Новый метод распараллеливания по итерациям для синтеза параллельно-конвейерных программ для РВС позволяет эффективно задействовать намного больший объем вычислительного ресурса по сравнению с методом распараллеливания по слоям. В частности, при обработке графа на 4000 вершин с вероятностью ребра 0,5 время решения задачи поиска максимальных клик по сравнению с методом распараллеливания по слоям сократилось в 10 раз. В то же время, как показано в работах [17, 18], метод распараллеливания по слоям быстрее реализаций для многопроцессорных систем. Таким образом, новый метод распараллеливания по итерациям позволяет обрабатывать графовые задачи большей размерности за приемлемое время и становится эффективней метода распараллеливания по слоям при увеличении количества доступного вычислительного ресурса и размерности решаемых графов.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Кирьянов А.Г., Ляхов А.И., Некрасов П.О., Платов Д.А. и др.* Протокол многоадресной маршрутизации Proximity-based Groupcast in MANET (GiM) // Информационные процессы. – 2012. – № 3. – Т. 12. – С. 213-228.
2. *Курейчик В.М., Глушань В.М., Щербаков Л.И.* Комбинаторные аппаратные модели и алгоритмы в САПР. – М.: Радио и связь, 1990. – 216 с.
3. *Jens Clausen.* Branch and bound algorithms principles and examples. Department of comp sc., university of Copenhagen, Universitetsparken 1, DK-2100 Corpenhagen, Denmark.(March 12, 1999).
4. *Кирюшин Н.К., Михалев И.В.* Использование многоядерных ускорителей для решения задачи пропозициональной выполнимости // Проблемы науки. – 2017. – № 22 (104).
5. *Dasari N.S., Ranjan D., Mohammad Z.* Maximal Clique Enumeration for Large Graphs on Hadoop Framework // Proc. of the First Workshop on Parallel Programming for Analytics Applications. – 2014. – P. 21-30. – Doi: 10.1145/2567634.2567640.
6. *Rossi R.A., Gleich D.F., Gebremedhin A.H.* Parallel Maximum clique Algorithms with Applications to Network Analysis // SIAM J. Sci. Comput. – 2015. – Vol. 37, No. 5. – P. 589-616. – Doi: 10.1137/14100018X.
7. *Kovács L., Szabó G.* Conceptualization with Incremental Bron-Kerbosch Algorithm in Big Data Architecture // Acta Polytechnica Hungarica. – 2016. – Vol. 13, No. 2. – P. 139-158. – Doi: 10.12700/aph.13.2.2016.2.8.
8. *Pattabiraman B. et al.* Fast Algorithms for the Maximum Clique Problem on Massive Graphs with Applications to Overlapping Community Detection // Internet Mathematics. – 2015. – Vol. 11, No. 4-5. – P. 421-448. – Doi: 10.1080/15427951.2014.986778.
9. *Dasari N.S., Zubair M., Ranjan D.* A Novel Parallel Algorithm for Maximal Clique Enumeration on Multicore and Distributed Memory Architectures. – 10 p. – URL: https://pdfs.semanticscholar.org/9827/9e2cedb14085886fcb4473f1ba483a3df195.pdf (дата обращения: 23.09.2020).
10. *Dasari N.S., Desh Z.M.* pbitMCE: A bit-based approach for maximal clique enumeration on multicore processors // in: Proc. 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS 2014). – 2014. – P. 478-485. – DOI: 10.1109/PADSW.2014.7097844.

11. *Bron C., Kerbosch J.* Algorithm 457: Finding All Cliques of an Undirected Graph // Comm of ACM. – 1973. – Vol. 16. – P. 575-577.
12. *Johnson D.J. and Trick M.A.* Eds., Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, Workshop, October 11-13, 1993. Boston, MA, USA: American Mathematical Society, 1996.
13. *Moon J. and Moser L.* On cliques in graphs // Israel Journal of Mathematics. – 1965. – Vol. 3, No. 1. – P. 23-28. Available: Doi: 10.1007/BF02760024.
14. *Каляев И.А. и др.* Реконфигурируемые мультиконвейерные вычислительные структуры / под общ. ред. И.А. Каляева. – 2-е изд., перераб. и доп. – Ростов-на-Дону: Изд-во ЮНЦ РАН, 2009. – 344 с.
15. Научно-исследовательский центр супер-ЭВМ и нейрокомпьютеров. – http://superevm.ru (дата обращения: 23.09.2020).
16. *Каляев А.В., Левин И.И.* Модульно-наращиваемые многопроцессорные системы со структурно-процедурной организацией вычислений. – М.: Янус-К, 2003. – 380 с.
17. *Касаркин А.В., Левин И.И.* Структурная реализация задачи нахождения всех максимальных клик графа на реконфигурируемых вычислительных системах // Вестник компьютерных и информационных технологий. – 2018. – № 10. – С. 3-10. – Doi: 10.14489/vkit.2018.10.pp.003-010.
18. *Касаркин А.В., Левин И.И.* Реализация алгоритма Брона-Кербоша на реконфигурируемых вычислительных системах // Известия ЮФУ. Технические науки. – 2019. – № 7 (209). – С. 142-152. – Doi: 10.23683/2311-3103-2019-7-142-152.
19. *Levin I., Dordopulo A., Fedorov A., Doronchenko Y.* Design Technology for Reconfigurable Computer Systems with Immersion Cooling. In: Voevodin V., Sobolev S. (eds.) Supercomputing. RuSCDays 2018. Communications in Computer and Information Science, Vol 965. Springer, Cham. – Doi: 10.1007/978-3-030-05807-4_47.
20. *Левин И.И., Пелипец А.В.* Эффективная реализация распараллеливания на реконфигурируемых системах // Вестник компьютерных и информационных технологий. – 2018. – № 8. – С. 11-16.
21. *Kasarkin A.V., Levin I.I., Sorokin D.A.* New iteration parallel-based method for solving graph NP-complete problems with reconfigurable computer systems // IOP Conf. Series: Materials Science and Engineering. – 2020. – Vol. 919 (1). – P. 052007(1-7). – Doi: 10.1088/1757-899X/919/5/052007.

REFERENCES

1. *Kir'yanov A.G., Lyahov A.I., Nekrasov P.O., Platov D.A. i dr.* Protokol mnogoadresnoj marshrutizacii Proximity-based Groupcast in MANET (GiM) [A multiaddress routing protocol Proximity-based Groupcast in MANET (GiM)], *Informacionnye process* [Informatsionnyie protsessy], 2012, No. 3, Vol. 12, pp. 213-228.
2. *Kurejchik V.M., Glushan' V.M., Shcherbakov L.I.* Kombinatornye apparatnye modeli i algoritmy v SAPR [Combinatory hardware models and algorithms for CAD]. Moscow: Radio i svyaz', 1990, 216 p.
3. *Jens Clausen.* Branch and bound algorithms principles and examples. Department of comp sc., university of Copenhagen, Universitetsparken 1, DK-2100 Corpenhagen, Denmark.(March 12, 1999).
4. *Kiryushin N.K., Mikhalev I.V.* Ispol'zovanie mnogoyadernykh uskoriteley dlya resheniya zadachi propozitsional'noy vypolnimosti [Use of multicore accelerators for tasks of propositional satisfiability], *Problemy nauki* [Problemy Nauki], 2017, No. 22 (104).
5. *Dasari N.S., Ranjan D., Mohammad Z.* Maximal Clique Enumeration for Large Graphs on Hadoop Framework, *Proc. of the First Workshop on Parallel Programming for Analytics Applications*, 2014, pp. 21-30. Doi: 10.1145/2567634.2567640.
6. *Rossi R.A., Gleich D.F., Gebremedhin A.H.* Parallel Maximum clique Algorithms with Applications to Network Analysis, *SIAM J. Sci. Comput.*, 2015, Vol. 37, No. 5, pp. 589-616. Doi: 10.1137/14100018X.
7. *Kovács L., Szabó G.* Conceptualization with Incremental Bron-Kerbosch Algorithm in Big Data Architecture, *Acta Polytechnica Hungarica*, 2016, Vol. 13, No. 2, pp. 139-158. Doi: 10.12700/aph.13.2.2016.2.8.

8. *Pattabiraman B. et al.* Fast Algorithms for the Maximum Clique Problem on Massive Graphs with Applications to Overlapping Community Detection, *Internet Mathematics*, 2015, Vol. 11, No. 4-5, pp. 421-448. Doi: 10.1080/15427951.2014.986778.
9. *Dasari N.S., Zubair M., Ranjan D.* A Novel Parallel Algorithm for Maximal Clique Enumeration on Multicore and Distributed Memory Architectures. 10 p. Available at: https://pdfs.semanticscholar.org/9827/9e2cedb14085886fcb4473f1ba483a3df195.pdf (accessed 23 September 2020).
10. *Dasari N.S., Desh Z.M.* pbitMCE: A bit-based approach for maximal clique enumeration on multicore processors, *in: Proc. 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS 2014)*, 2014, pp. 478-485. Doi: 10.1109/PADSW.2014.7097844.
11. *Bron C., Kerbosch J.* Algorithm 457: Finding All Cliques of an Undirected Graph, *Comm of ACM*, 1973, Vol. 16, pp. 575-577.
12. *Johnson D.J. and Trick M.A.* Eds., Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, Workshop, October 11-13, 1993. Boston, MA, USA: American Mathematical Society, 1996.
13. *Moon J. and Moser L*. On cliques in graphs, *Israel Journal of Mathematics*, 1965, Vol. 3, No. 1, pp. 23-28. Available: Doi: 10.1007/BF02760024.
14. *Kalyaev I.A. i dr.* Rekonfiguriruemye mul'tikonveyernye vychislitel'nye struktury [Reconfigurable multipipeline computing structures], under general ed. of I.A. Kalyaev. 2nd ed., revised and enlarged. Rostov-on-Don: Izd-vo YUNTS RAN, 2009, 344 p.
15. Nauchno-issledovatel'skiy tsentr super-EVM i neyrokomp'yuterov [Scientific research center of supercomputers and neurocomputers]. Available at: http://superevm.ru (accessed 23 September 2020).
16. *Kalyaev A.V., Levin I.I.* Modul'no-narashchivaemye mnogoprotsessornye sistemy so strukturno-protsedurnoy organizatsiey vychisleniy [Modular-scalable multiprocessor systems with structural and procedural organization of calculations]. Moscow: Yanus-K, 2003, 380 p.
17. *Kasarkin A.V., Levin I.I.* Strukturnaya realizatsiya zadachi nakhozhdeniya vsekh maksimal'nykh klik grafa na rekonfiguriruemykh vychislitel'nykh sistemakh [Structural implementation of the task of maximal clique enumeration on reconfigurable computer systems], *Vestnik komp'yuternykh i informatsionnykh tekhnologiy* [Herald of computer and information technologies], 2018, No. 10, pp. 3-10. Doi: 10.14489/vkit.2018.10.pp.003-010.
18. *Kasarkin A.V., Levin I.I.* Realizatsiya algoritma Brona-Kerbosha na rekonfiguriruemykh vychislitel'nykh sistemakh [Implementation of the Bron-Kerbosch algorithm on reconfigurable computer systems], *Izvestiya YuFU. Tekhnicheskie nauki* [Izvestiya SFedU. Engineering sciences], 2019, No. 7 (209), pp. 142-152. Doi: 10.23683/2311-3103-2019-7-142-152.
19. *Levin I., Dordopulo A., Fedorov A., Doronchenko Y.* Design Technology for Reconfigurable Computer Systems with Immersion Cooling. In: Voevodin V., Sobolev S. (eds.) Supercomputing. RuSCDays 2018. Communications in Computer and Information Science, Vol 965. Springer, Cham. Doi: 10.1007/978-3-030-05807-4_47.
20. *Levin I.I., Pelipets A.V.* Effektivnaya realizatsiya rasparallelivaniya na rekonfiguriruemykh sistemakh [Efficient Parallel Execution on Reconfigurable Systems], *Vestnik komp'yuternykh i informatsionnykh tekhnologiy* [Herald of computer and information technologies], 2018, No. 8, pp. 11-16.
21. *Kasarkin A.V., Levin I.I., Sorokin D.A.* New iteration parallel-based method for solving graph NP-complete problems with reconfigurable computer systems, *IOP Conf. Series: Materials Science and Engineering*, 2020, Vol. 919 (1), pp. 052007(1-7). Doi: 10.1088/1757-899X/919/5/052007.

Статью рекомендовал к опубликованию д.т.н. Э.В. Мельник.

Касаркин Алексей Викторович – Южный федеральный университет; e-mail: kav589@mail.ru; 347931, г. Таганрог, пер. Итальянский, 106; тел.: +79045065636; кафедра ИМС, аспирант.

Kasarkin Alexey Viktorovich – Southern Federal University; e-mail: kav589@mail.ru; 106, Italyansky lane, Taganrog, 347931, Russia; phone: +79045065636; graduate student.