Болдырихин Николай Вячеславович – Донской государственный технический университет; e-mail: boldyrikhin@mail.ru; 344065, г. Ростов-на-Дону, пер. Днепровский, 116К, кв. 111; тел.: +79043442295; кафедра кибербезопасности информационных систем; к.т.н.; доцент.

Короченцев Денис Александрович – e-mail: mytelefon@mail.ru; 344038, г. Ростов-на-Дону, пр. Михаила Нагибина, 29, кв. 24; тел.: +79034895173; кафедра кибербезопасности информационных систем; зав. кафедрой; к.т.н.

Алтунин Федор Александрович – ООО «Яндекс.Маркет Лаб»; email: altuninf@gmail.com; Москва, Ленинский пр-т 129, корп. 3, кв. 119; тел.: +79889496866; инженер по тестированию.

Boldyrikhin Nikolay Vyacheslavovich – Don State Technical University; e-mail: boldyrikhin@mail.ru; 116K, per. Dneprovsky, apt. 111, Rostov-on-Don, 344065, Russia; phone: +79043442295; the department of cybersecurity of information systems; cand. of eng. sc.; associate professor.

Korochentsev Denis Aleksandrovich – e-mail: mytelefon@mail.ru; 29, Mikhail Nagibin Ave., apt. 24, Rostov-on-Don, 344038, Russia; phone: +79034895173; the department of cybersecurity of information systems; head of the department; cand. of eng. sc.

Altunin Fedor Aleksandrovich – Yandex.Market Lab LLC; e-mail: altuninf@gmail.com; 129, Leninsky Prospect building. 3, apt. 119, Moscow, Russia; phone: +79889496866; testing engineer.

УДК 004.432.4 DOI 10.18522/2311-3103-2020-3-98-111

И.И. Левин, А.И. Дордопуло, И.В. Писаренко, Д.В. Михайлов

ПРЕДСТАВЛЕНИЕ ГРАФОВ С АССОЦИАТИВНЫМИ ОПЕРАЦИЯМИ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ SET@L

Как правило, информационный граф с ассоциативными операциями реализуется в виде последовательной («голова/хвост») или параллельной («разбиение пополам») топологии, причем обе структуры содержат одинаковое число операционных вершин. Редукционные преобразования графов с представленными топологиями при недостатке вычислительного ресурса не обеспечивают создание эффективной ресурсонезависимой программы: вариант «разбиение пополам» характеризуется нерегулярной межитерационной коммутацией, а структура «голова/хвост» – увеличенной скважностью данных при редукции. В данной статье предлагается преобразовать топологию графа с ассоциативными операциями в один из комбинированных вариантов с последовательными и параллельными фрагментами вычислений, синтезированный в соответствии с заданным вычислительным ресурсом. Это позволяет повысить удельную производительность вычислений при редукции. Модифицированная топология включает изоморфные подграфы с топологией «разбиение пополам», содержащие максимальное число аппаратно реализуемых операционных вершин, а обработка промежуточных данных осуществляется по принципу «голова/хвост». Вычислительная структура для рассмотренной топологии имеет минимальную латентность и состоит из одного базового подграфа и одной вершины, в которую редуцируется блок обработки промежуточных данных с топологией «голова/хвост». Разработан алгоритм, позволяющий в зависимости от доступного аппаратного ресурса перейти от базового последовательного варианта реализации к различным комбинированным топологиям вплоть до предельного случая топологии «разбиение пополам». Поскольку традиционные методы параллельного программирования могут описать множество топологий только в виде набора отдельных подпрограмм, для создания ресурсонезависимого описания графов с ассоциативными операциями предлагается использовать язык архитектурно-независимого программирования Set@l. Принципы построения топологий «голова/хвост» и «разбиение пополам» описаны в виде признаков

метода обработки множеств на языке Set@l, а ресурсонезависимая программа оперирует этими типами и типами параллелизма для модификации топологии графа и последующей редукции производительности в соответствующих аспектах программы.

Информационные графы с ассоциативными операциями; ресурсонезависимое программирование; редукция производительности; язык программирования Set@l; признаки «голова/хвост» и «разбиение пополам».

I.I. Levin, A.I. Dordopulo, I.V. Pisarenko, D.V. Mihaylov

DESCRIPTION OF GRAPHS WITH ASSOCIATIVE OPERATIONS IN SET@L PROGRAMMING LANGUAGE

Usually, an information graph with associative operations has a sequential ("head/tail") or parallel ("half-splitting") topology with invariable quantity of operational vertices. If computational resource is insufficient for the implementation of all vertices, the reduction transformations of graphs with basic topologies do not allow for the creation of an efficient resource-independent program. In fact, the "half-splitting" variant is characterized by irregular connections between iterations, and the "head/tail" structure has an increased data duty cycle in the reduced form. In this paper, we propose to transform the topology of a graph with associative operations into a combined variant with sequential and parallel fragments of calculations. The resultant combined topology depends on computational resource of a parallel computer system, and such transformation provides the improvement of specific performance for the reduced computing structure. The considered topology contains isomorphic subgraphs with the "half-splitting" topology, which include the maximal number of hardwarily implemented operational vertices, but the processing of intermediate data is performed using the "head/tail" principle. The computing structure for the combined topology has minimal latency and includes one basic subgraph and one vertex with feedback. This vertex is obtained as a result of the "head/tail" block reduction. We develop an algorithm for the conversion of the initial sequential graph to various combined topologies or to the limiting case of the "half-splitting" topology with regard to available hardware resource. Within traditional methods of parallel programming, it is possible to describe the variety of topologies only as a set of separated subprograms. To create an efficient resource-independent program, we propose the application of the Set@l programming language. We describe the "head/tail" and "half-splitting" principles as the attributes of set processing methods in Set@l. Resource-independent program uses these types and parallelism attributes for the modification of topology and further reduction of performance in the corresponding aspects.

Information graphs with associative operations; resource-independent programming; performance reduction; Set@l programming language; "head/tail" and "half-splitting" attributes.

Введение. Ассоциативность представляет собой фундаментальное свойство бинарных операций, которое определяет независимость результата расчетов от порядка вычислений [1, 2]. Типовыми графами на основе ассоциативных операций являются графы сложения и умножения (при выполнении определенных условий), конъюнкции и дизъюнкции элементов массива, поиска максимума и минимума. Известны два базовых варианта топологии таких графов с одинаковым числом вершин: последовательный (линейная структура) и параллельный (пирамидальная или каскадная структура) [3–5]. В действительности, помимо двух указанных случаев существует множество комбинированных вариантов топологий, составленных из чередующихся параллельных и последовательных фрагментов вычислений [6].

Для масштабирования параллельных вычислений при решении прикладных задач на вычислительных системах (ВС) с реконфигурируемой архитектурой [7–10] используются методы редукции производительности [11]. Редукционные преобразования информационных графов на основе ассоциативных операций с последовательной и параллельной топологиями не обеспечивают создание эффективной ресурсонезависимой программы: граф с пирамидальной структурой характеризуется нерегулярной межитерационной коммутацией, а последовательный

вариант реализации имеет самую большую латентность, которая приводит к существенному увеличению скважности данных при редукции. Поскольку все операции в рассматриваемых графах ассоциативны, можно модифицировать топологию графа для удобства редукционных преобразований. Преобразование графа в кадр [12] возможно только при его изоморфной структуре, поэтому необходимо использовать один из комбинированных вариантов топологии, который содержит изоморфные подграфы. Для заданного вычислительного ресурса возможно синтезировать информационно эквивалентный граф, содержащий аппаратно реализуемые изоморфные подграфы с максимальной степенью параллелизма ассоциативных операций. Традиционные методы программирования для реконфигурируемых и гибридных ВС [13] не позволяют изменять информационный граф решаемой прикладной задачи в зависимости от доступного вычислительного ресурса, поэтому каждая топология может быть описана только в виде отдельной подпрограммы.

В работах [14–16] предложен язык архитектурно-независимого параллельного программирования Set@l (Set Aspect-Oriented Language), основанный на парадигме аспектно-ориентированного программирования (АОП) и теоретико-множественном представлении исходного кода программы. Описав базовые принципы построения графов с ассоциативными операциями в виде специальных признаков метода обработки на языке Set@l, возможно синтезировать множество вариантов топологий и переходить между ними путем изменения типов и разбиений совокупностей. Приведение графа к редуцируемому виду при заданном объеме вычислительного ресурса реконфигурируемой ВС может осуществляться без изменения исходного кода программы.

Преобразование графов с ассоциативными операциями для эффективной редукции производительности. Информационные графы, которые состоят из двуместных ассоциативных операций f с одним выходом (например, операций сложения, умножения, конъюнкции или дизъюнкции, поиска максимума или минимума), могут быть построены по одному из двух базовых принципов [3–5]. Примеры стандартных топологий таких графов для операционных вершин с двумя входами изображены на рис. 1. Принцип «*голова/хвост*» (рис. 1,а) предполагает последовательное выполнение ассоциативной операции f над элементами a_i множества входных данных A. На каждой итерации вычислений операция f выполняется над одним из элементов входных данных a_i («*головой*») [17] и промежуточным элементом s_j, который представляет собой результат обработки оставшейся части данных («*хвоста*») [17]. Исключением является итерация, на которой обрабатываются сразу два элемента входных данных a_1 и a_2. На выходе каждой из операций формируется промежуточный (s_j) или окончательный (*Res*) результат вычислений.

Другой принцип – «*разбиение* (*деление) пополам*» (рис. 1,б) – предполагает параллельное выполнение ассоциативных операций f и основан на известном подходе «разделяй и властвуй», получившем широкое распространение в программировании [18]. Если двигаться от выходной вершины графа к его входным вершинам, то на каждой итерации множество входных данных $A=\{a_1, a_2, \dots, a_8\}$ (на первой итерации) или все его подмножества (на остальных итерациях) делятся на два подмножества A_1 и A_2, каждое из которых содержит одинаковое число обрабатываемых элементов. Промежуточные результаты s_i и s_{i+1} обработки подмножеств-половин A_1 и A_2 являются входными данными для операции f на текущей итерации. Разбиение и движение по графу продолжается до тех пор, пока не будет получено по два элемента в каждом подмножестве исходного множества A.

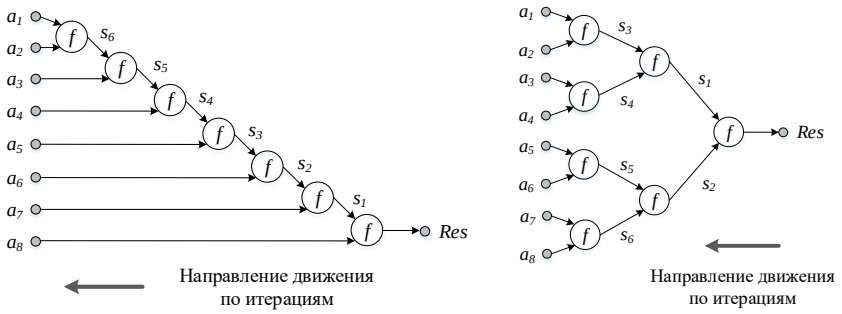


Рис. 1. Топологии графов на основе двуместных ассоциативных операций f, построенные по принципам «голова/хвост» (а) и «деление (разбиение) пополам» (б)

Представленные на рис. 1 топологии графов содержат одинаковое количество операционных вершин, но имеют разную латентность вычислительных структур: $\tau = n - 1$ для топологии «голова/хвост» (рис. 1,а) и $\tau = \log_2 n$ для топологии «разбиение пополам» (рис. 1,б), где n – мощность множества входных данных A.

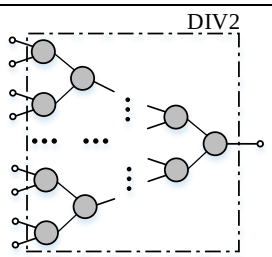
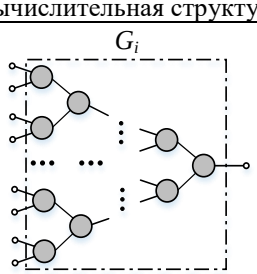
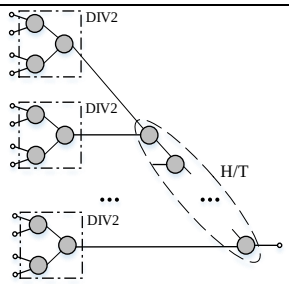
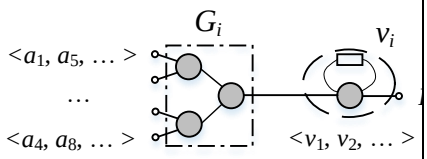
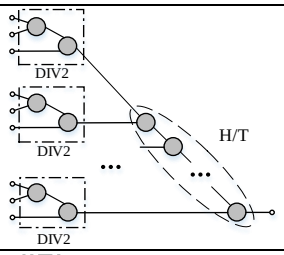
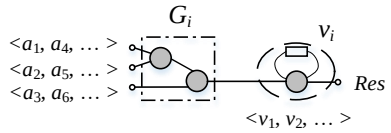
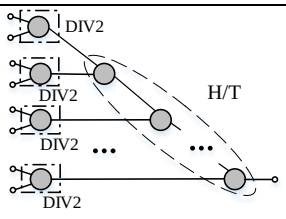
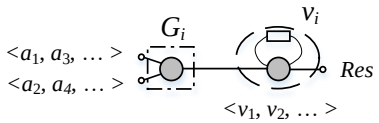
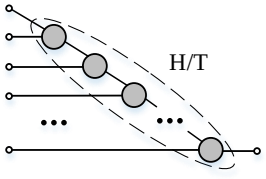
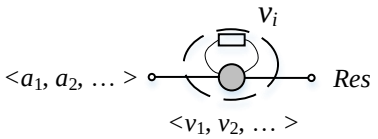
Если доступного вычислительного ресурса реконфигурируемой ВС достаточно для аппаратной реализации всех операционных вершин информационного графа с ассоциативными операциями, то используется топология с наименьшей латентностью – «разбиение пополам» (случай $R/R_0=n–1$ в табл. 1, где R – объем доступного вычислительного ресурса, R_0 – ресурс, занимаемый одной операционной вершиной). При нехватке вычислительного ресурса необходимо масштабировать вычисления с помощью методов редукции производительности [11]. Однако описание редукционных преобразований для топологии «разбиение пополам» достаточно трудоемко и громоздко из-за нерегулярной межитерационной коммутации и зависимости от размерности решаемой задачи. Свойство ассоциативности базовых операций позволяет модифицировать информационный граф задачи и перейти к топологии «голова/хвост», которая имеет удобную для редукции регулярную коммутационную структуру. С другой стороны, если невозможно разместить все операционные вершины графа, структурно-процедурная реализация топологии «голова/хвост» редуцируется до одной вершины с обратной связью ($R/R_0=1$ в табл. 1) [12], что обеспечивает высокую удельную производительность только в случае минимального объема доступного вычислительного ресурса R_0. При конфигурациях вычислительного ресурса $R/R_0=2÷(n–2)$ необходимо перейти к одному из комбинированных вариантов топологии, содержащему последовательные и параллельные фрагменты вычислений [6], что позволит повысить удельную производительность. Прежде комбинированные топологии не рассматривались, так как в большинстве своем они не обладают регулярной и изоморфной структурой, что затрудняет автоматическое масштабирование вычислений и редукцию производительности. Поэтому ресурсонезависимое описание графа с ассоциативными операциями должно учитывать не только два предельных случая «разбиение пополам» и «голова/хвост», но и множество комбинированных вариантов топологий и правила перехода между ними для редукции производительности при разных конфигурациях ВС.

Топология информационного графа с ассоциативными операциями, модифицированная для редукции производительности при заданном объеме вычислительного ресурса реконфигурируемой ВС, включает две составляющие. Поскольку преобразование графа в кадр возможно только при его изоморфной структуре [12], первой составляющей являются изоморфные подграфы DIV2 (табл. 1), которые построены по принципу «разбиение пополам» и содержат максимальное число аппа-

ратно реализуемых операционных вершин. Вторая составляющая представляет собой блок Н/Т, построенный по принципу «голова/хвост» (табл. 1) и необходимый для обработки промежуточных данных и вычисления конечного результата.

Таблица 1

Топологии информационных графов с ассоциативными операциями и соответствующие им вычислительные структуры при разных объемах доступного вычислительного ресурса *R* реконфигурируемой ВС

Конфигурация	Топология	Вычислительная структура
$R/R_0=n-1$		
…		
$R/R_0=4$		
$R/R_0=3$		
$R/R_0=2$		
$R/R_0=1$		

Если скважность данных при обратной связи составляет один такт, то блок Н/Т редуцируется до одной операционной вершины [6]. Поэтому аппаратно реализуемая вычислительная структура (см. табл. 1) включает один базовый подграф G_i и одну вершину v_i, которая соответствует редуцированному блоку Н/Т. Если вычислительного ресурса хватает для реализации только одной операционной вершины (R/R_0=1 в табл. 1), то комбинированная топология преобразуется в классическую топологию «голова/хвост»: подграфы DIV2 не реализуются, а единственная в вычислительной структуре операционная вершина соответствует блоку Н/Т размерностью (n–1). В другом предельном случае, когда вычислительного ресурса ВС достаточно для размещения всех вершин информационного графа (R/R_0=(n–1) в табл. 1), комбинированная топология переходит в стандартную топологию «разбиение пополам»: единственный подграф DIV2 включает все операционные вершины.

Исходный информационный граф с топологией «голова/хвост» возможно преобразовать в граф с одной из комбинированных топологий (см. табл. 1) или топологией «разбиение пополам» в зависимости от доступного вычислительного ресурса ВС R следующим образом:

1. Подготовительный этап: разделим все операционные вершины исходного графа на r групп по k вершин в каждой, где r – коэффициент редукции, k=**floor**(R/R_0)–1. Между соседними группами оставим по одной вершине для обработки промежуточных данных, как показано на рис. 2. Если в последнюю группу войдет меньше, чем k вершин, то дополним ее операционными вершинами до требуемой размерности.

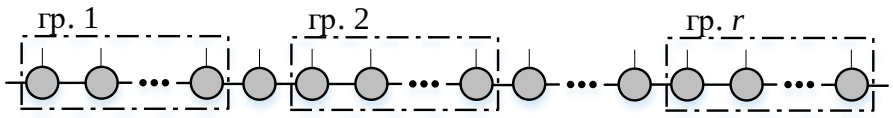
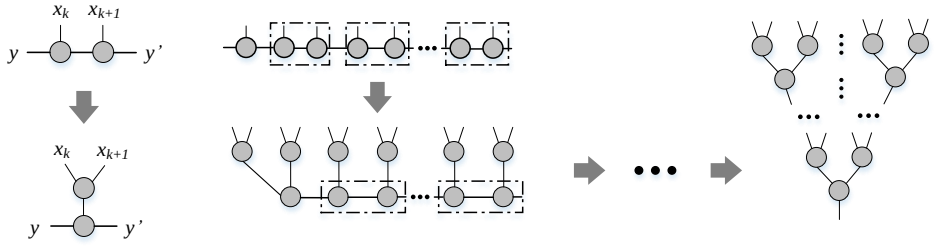


Рис. 2. Выделение групп операционных вершин в исходном графе, построенном по принципу «голова/хвост»

2. Формирование изоморфных подграфов с топологией «разбиение пополам»: выделяя пары соседних вершин и изменяя порядок выполнения операций в каждой паре в соответствии с рис. 3,а, преобразуем топологию каждой группы операционных вершин к топологии «разбиение пополам». Последовательность преобразований для одной из групп показана на рис. 3,б. В результате получим информационный граф, показанный на рис. 3,в.

3. Объединение подграфов с топологией «разбиение пополам» по принципу «голова/хвост»: Изменим порядок выполнения операций для всех пар «отдельная вершина – группа» по принципу, изображенному на рис. 4,а. После преобразования получим конечную топологию информационного графа, показанную на рис. 4,б.



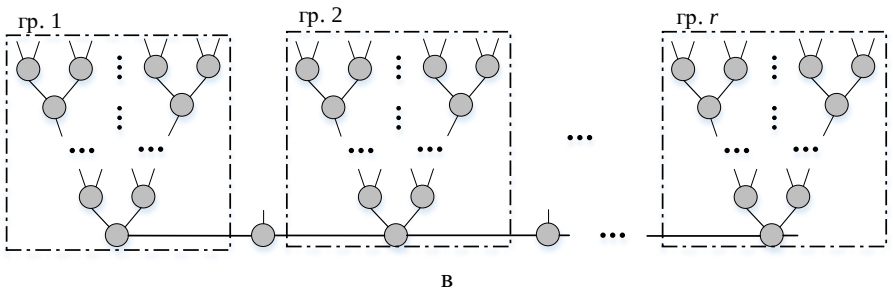


Рис. 3. Преобразование пары операционных вершин (а), последовательность преобразований в одной группе (б) и топология информационного графа (в), получаемая после преобразования всех групп операционных вершин исходного графа по принципу «разбиение пополам»

При структурно-процедурной реализации информационно-независимые изоморфные подграфы G_1, G_2, …, G_r (см. рис. 4,б), выделенные в результате модификации топологии графа с ассоциативными операциями, преобразуются в подкадр G_i [19] (рис. 4,в), на входы которого в кадре подаются кортежи данных. Из блока G_{r+1}, содержащего информационно-зависимые операционные вершины v_1, v_2, …, v_{r-1}, формируется одна дополнительная вершина с обратной связью v_i.

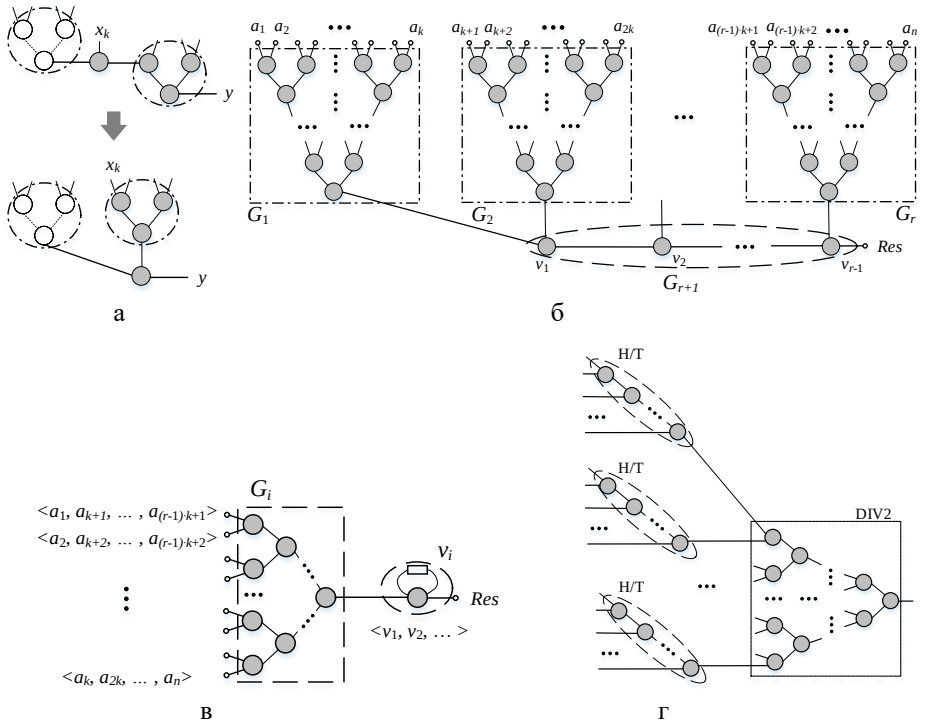


Рис. 4. Преобразование пары «отдельная вершина – группа» (а), конечная комбинированная топология графа (б) и соответствующая ей вычислительная структура (в), комбинированная топология с изоморфными подграфами, построенными по принципу «голова/хвост» (г)

Следует отметить, что для эффективной реализации структурно-процедурной организации вычислений представляет интерес и другой вариант комбинированной топологии графа с ассоциативными операциями, показанный на рис. 4, г. Если латентность операционной вершины превышает один такт, то блоки последовательной обработки H/T преобразуются в независимые устройства с накоплением, которые объединяются по схеме «конвейер в конвейере» [19]. Данный подход будет рассмотрен более подробно в последующих работах.

Описание признаков «голова/хвост» и «разбиение пополам» на языке программирования Set@l. Традиционные методы параллельного программирования, как правило, оперируют информационным графом с фиксированной структурой, поэтому их использование для описания преобразований топологии графа в соответствии с предложенным алгоритмом (см. рис. 2–4) достаточно трудоемко и малоэффективно. Код ресурсонезависимой программы, которая реализует рассмотренные преобразования, будет состоять из множества подпрограмм, связанных условными операторами и задающих отдельные варианты топологий. В отличие от классических языков параллельного программирования, возможности языка архитектурно-независимого программирования Set@l позволяют описать принципы построения графов с ассоциативными операциями в виде специальных признаков метода обработки, присваиваемых обрабатываемому множеству *A*. В таком случае исходный код программы описывает не отдельные реализации, а множество возможных топологий графа для заданной размерности задачи. Аспекты выбирают определенный вариант топологии, исходя из конкретных значений параметров конфигурации. Для изменения структуры информационного графа достаточно изменить тип и разбиение множества *A*, тогда как исходный код программы остается неизменным.

Рассмотрим описание принципов «голова/хвост» и «разбиение пополам» на языке программирования Set@l для стандартных топологий информационных графов с ассоциативными операциями, изображенных на рис. 1. С помощью синтаксической конструкции `attribute` [20] введем дополнительные признаки, код которых приведен на рис. 5.

```
attribute [f(a,b,c)|type(f)=Op]:
   operand(element(a,b,c),attribute(Op));
   c=f(a,b);
end(Op2);
```

```
attribute Head(A):
   operand(set(A),element(Head(A)));
   Head(A)=A(1);
end(Head);
```

```
attribute Tail(A):
   operand(set(A),set(Tail(A)));
   Tail(A)=dif(A,A(1));
end(Head);
```

```
attribute d2(A,A1,A2):
   operand(set(A,A1,A2));
   n=card(A);
   A1=(A(k)|k<=n/2);
   A2=dif(A,A1);
end(d2);
```

Рис. 5. Описания признаков бинарной операции (`f`), выделения «головы» (`Head`) и «хвоста» (`Tail`) множеств и разбиения совокупности на два равномощных подмножества (`d2`) на языке программирования Set@l

Признак базовой операции `f` (см. рис. 5) описан в обобщенном виде: конкретный тип операции `Op` уточняется в другом модуле программы и может принимать разные значения (например, `'+'` или `'*'`). Директива `operand` указывает типы объектов, которым может быть присвоен тот или иной признак.

Описание принципа построения топологии «голова/хвост» (`H/T`, граф на рис. 1,а) на языке программирования Set@l представлено на рис. 6. Признак линейки операций `Lf` строится рекурсивно (`Rec`) с использование введенного ранее

признака базовой бинарной операции f (см. код на рис. 5) и задает отношение между обрабатываемым множеством данных A и результирующим элементом Res. Граф строится в направлении от выходной вершины к входным. На каждой итерации линейку операций Lf над элементами множества A можно представить как объединение линейки операций Lf над «хвостом» множества A и отдельной вершины f, на входы которой подаются «голова» множества A и промежуточный результат s последовательного выполнения ассоциативных операций f над Tail(A), а выход Res является конечным или промежуточным результатом вычислений (см. строку 5 на рис. 6). Синтаксическая конструкция break[<условие>:<операция>] (строка 4 на рис. 6) выделяет условие завершения рекурсии и описывает операцию, завершающую формирование структуры информационного графа. При выполнении условия последняя линейка операций Lf преобразуется в особую вершину, на входы которой подаются два оставшихся элемента множества A.

```
(1)  attribute [Lf(A,Res)|Lf=Rec(f),type(A)='H/T']:
(2)     operand(set(A),element(Res));
(3)     element(s);
(4)     Lf(A,Res)=break[card(Tail(A))=1:
(5)     f(Head(Tail(A)),Head(A),Res)],
(6)                  union[Lf(Tail(A),s),f(s,Head(A),Res)];
     end(Lf);
```

Рис. 6. Код признака Lf на языке программирования Set@l, реализующего принцип «голова/хвост» (H/T) для построения информационного графа с ассоциативными операциями f

Применяя аналогичные рассуждения и рекурсивный подход к описанию, можно описать параллельный принцип построения графов «разбиение пополам» (DIV2, граф на рис. 1,б) на языке программирования Set@l. В коде, приведенном на рис. 7, граф описан в направлении от выходной вершины к входным. На каждой итерации исходное множество A разбивается признаком d2 на два подмножества A1 и A2 с равным количеством элементов (строка 3 на рис. 7). В таком случае пирамиду операций Pf над элементами A можно представить как объединение пирамиды операций Pf над элементами подмножества A1, пирамиды операций Pf над элементами подмножества A2 и отдельной вершины f (строка 6 на рис. 7). На входы этой вершины f подаются промежуточные результаты s1, s2 выполнения пирамиды операций над Pf подмножествами совокупности A, а выход Res представляет собой конечный или промежуточный результат вычислений. Рекурсия завершается при выполнении условия, показанного в строке 5 на рис. 7. Распараллеливание вычислений достигается путем удвоения числа веток рекурсии на каждом шаге преобразований.

Таким образом, язык архитектурно-независимого программирования Set@l позволяет описать базовые принципы построения графов на основе ассоциативных операций с одним выходом в виде специальных признаков метода обработки H/T и DIV2, которые присваиваются множеству входных данных A. В отличие от предложенных ранее типов множеств по параллелизму par, seq, pipe, conc и imp [15, 16], данные признаки определяют не конкретные методы распараллеливания вычислений, а общую структуру информационного графа решаемой прикладной задачи, которая может быть модифицирована в соответствии с конкретной архитектурой или конфигурацией параллельной ВС. Однократно описав

принципы «голова/хвост» и «разбиение пополам», возможно получить множество разных вариантов реализации вычислений без изменения исходного кода программы. Некоторые примеры фрагментов кода, использующих типы обработки H/T и DIV2 для синтеза различных информационных графов, приведены на рис. 8.

```
(1)  attribute [Pf(A,Res)|Pf=Rec(f),type(A)='DIV2')]:
(2)     operand(set(A),element(Res));
(3)     d2(A,A1,A2);
(4)     element(s1,s2);
(5)     Pf(A,Res)=break[card(A1)=1 and card(A2)=1: f(Head(A1),Head(A2),Res)],
(6)                  union[Pf(A1,s1),Pf(A2,s2),f(s1,s2,Res)];
(7)  end(Pf);
```

Рис. 7. Код признака Pf на языке программирования Set@l, который реализует принцип «разбиение пополам» (DIV2) для описания информационного графа на основе ассоциативной операции f

Структура информационного графа G (см. код на рис. 8) определяется отношением Gf между обрабатываемым множеством A и результатом вычислений Res. Пользователю достаточно изменить только тип совокупности A для получения информационного графа с совершенно иной коммутационной структурой, тогда как обобщенные описания признаков «голова/хвост» и «разбиение пополам» остаются неизменными. Тип базовой ассоциативной операции f определяет функциональность операционных вершин синтезируемого информационного графа.

Линейка сумматоров	*Пирамида сумматоров*
G=**Gf**(A,Res); **Gf**=(**Rec**(**f**),**type**(A)='**H/T**'); **type**(f)='**+**';	G=**Gf**(A,Res); **Gf**=(**Rec**(**f**),**type**(A)='**DIV2**'); **type**(f)='**+**';
Линейка умножителей	*Пирамидальный поиск максимума*
G=**Gf**(A,Res); **Gf**=(**Rec**(**f**),**type**(A)='**H/T**'); **type**(f)='***';	G=**Gf**(A,Res); **Gf**=(**Rec**(**f**),**type**(A)='**DIV2**'); **type**(f)='**max**';

Рис. 8. Фрагменты кода, использующие типы совокупностей «голова/хвост» (H/T) и «разбиение пополам» (DIV2) для описания разных информационных графов на основе ассоциативных операций с одним выходом

Разработка ресурсонезависимой программы на языке Set@l. Используя признаки метода обработки H/T и DIV2 (см. программный код на рис. 6 и 7), возможно описать преобразование топологии информационного графа с ассоциативными операциями в соответствии с объемом доступного вычислительного ресурса (табл. 1) как изменение типизации и разбиения множества обрабатываемых данных *A*. В общем случае совокупность *A* должна иметь следующую форму, которая обеспечит выделение изоморфных подграфов, минимальную латентность вычислительной структуры и удобство дальнейших редукционных преобразований:

$$A = \mathbf{H/T}\,[subA_1, subA_2, \ldots, subA_r]; \quad (1)$$

$$subA_p = \mathbf{DIV2}\{a_b, a_{b+1}, \ldots, a_c\}, \quad (2)$$

где $subA_p$ – p-е подмножество (тип DIV2) множества A (тип H/T); r – коэффициент редукции производительности; b, c – границы диапазона индексов элементов множества A, попадающих в подмножество $subA_p$. При $r = 1$ формулы (1), (2) опи-

сывают топологию «разбиение пополам»: единственное подмножество $subA_1$ содержит все элементы $a_1 \ldots a_n$. Если $r = n–1$, то множество A состоит из n подмножеств с одним элементом, что соответствует топологии «голова/хвост».

Признак `graph_modification`, преобразующий исходную совокупность обрабатываемых данных `Array` в результирующее множество `A` с параметризованной структурой, определяемой формулами (1) и (2), описывается по тому же принципу, что и при редукционных преобразованиях графов сортирующих сетей в работе [14]. Фрагмент программного кода на рис. 9 выбирает ту или иную топологию информационного графа с ассоциативными операциями в зависимости от доступного вычислительного ресурса `R` реконфигурируемой ВС, размерности обрабатываемого массива `n` и ресурс `R0`, занимаемого одной операционной вершиной. Предложенный подход позволяет синтезировать топологию, которая наиболее удобна для дальнейших редукционных преобразований при заданном объеме вычислительного ресурса. В предельных случаях комбинированная топология переходит в базовые топологии «голова/хвост» или «разбиение пополам». В то же время обеспечивается минимальная латентность вычислительной структуры как при структурной, так и при структурно-процедурной реализации вычислений.

```
Q=floor(R/R0); // определение числа вершин, реализуемых на доступном
               // вычислительном ресурсе R;
r=ceil(n/Q);                    // расчет коэффициента редукции;
graph_modification(Array,A,r); // разбиение и типизация множества A;
```

Рис. 9. Программный код на языке Set@l, осуществляющий преобразование топологии информационного графа с ассоциативными операциями в соответствии с объемом доступного вычислительного ресурса

В результате работы транслятора языка программирования Set@l из совокупности `A` (см. формулы (1) и (2)) синтезируется множество `G`, которое описывают топологию информационного графа с ассоциативными операциями в следующей теоретико-множественной форме:

$$G = \vec{\{}\{subG_1, subG_2, \ldots, subG_r\}, v_1, v_2, \ldots, v_{r-1}\vec{\}}, \quad (3)$$

где $\vec{\{}\ \vec{\}}$ – обозначение параллельно-зависимого типа обработки (тип по параллелизму `conc` в языке программирования Set@l [9, 10]); { } – обозначение параллельно-независимого типа обработки (тип по параллелизму `par` [9, 10]); $subG_i$ – i-й подграф, построенный по принципу «разбиение пополам» (DIV2 в табл. 1); v_i – i-я операционная вершина, входящая в блок обработки промежуточных результатов, который построен по принципу «голова/хвост» (H/T в табл. 1). Дальнейшая редукция производительности осуществляется с помощью специального аспекта `reduction`, который преобразует совокупность (3) к следующему виду:

$$\mathrm{Gr} = < subG_1, \vec{\{}subG_2, v_1\vec{\}}, \vec{\{}subG_3, v_2\vec{\}}, \ldots, \vec{\{}subG_r, v_{r-1}\vec{\}} >. \quad (4)$$

Элементы исходного множества G перегруппируются таким образом, чтобы аппаратно реализуемая вычислительная структура содержала один подграф с топологией «разбиение пополам» и одну дополнительную вершину, которая соответствует последовательному блоку обработки промежуточных результатов H/T в табл. 1. Получаемая структура занимает весь доступный вычислительный ресурс R и имеет минимальную латентность.

Заключение. Таким образом, стандартные топологии информационных графов с ассоциативными операциями эффективны только в предельных случаях минимального и максимального объема доступного вычислительного ресурса рекон-

фигурируемой ВС. В остальных случаях целесообразно модифицировать топологию графа в соответствии с предложенным в статье алгоритмом и преобразовать ее к виду, удобному для проведения редукции производительности. Однако традиционные методы параллельного программирования оперируют информационным графом с фиксированной структурой, поэтому их использование для описания преобразований топологии графа в соответствии с параметрами конфигурации ВС достаточно трудоемко и малоэффективно. Для решения данной проблемы предложено использовать язык архитектурно-независимого программирования Set@l, в котором все возможные топологии графов с ассоциативными операции могут быть представлены в виде двух признаков метода обработки, присваиваемых множеству исходных данных, и их комбинаций. Разработана ресурсонезависимая программа на языке Set@l, которая включает аспект преобразования топологии графа и аспект редукции производительности.

Поддержка. Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 20-07-00545.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Кнут Д.Э.* Искусство программирования. Т. 4, А. Комбинаторные алгоритмы. Ч. 1: пер. с англ. – М.: ООО «И. Д. Вильямс», 2013. – 960 с.
2. *Новиков Ф.* Дискретная математика. – 3-е изд. – СПб.: Питер, 2019. – 496 с.
3. *Карепова Е.Д.* Основы многопоточного и параллельного программирования. – Красноярск: Сиб. федер. ун-т, 2016. – 356 с.
4. Задача суммирования элементов массива // Лаборатории Параллельных информационных технологий НИВЦ МГУ. – URL: https://parallel.ru/fpga/Summ2 (дата обращения: 27.04.2020).
5. *Старченко А.В., Берцун В.Н.* Методы параллельных вычислений. – Томск: Изд-во Том. ун-та, 2013. – 223 с.
6. *Ефимов С.С.* Обзор методов распараллеливания алгоритмов решения некоторых задач вычислительной дискретной математики // Математические структуры и моделирование. – 2007. – Вып. 17. – С. 72-93.
7. *Каляев И.А., Левин И.И., Семерников Е.А., Шмойлов В.И.* Развитие отечественных многокристалльных реконфигурируемых вычислительных систем: от воздушного к жидкостному охлаждению // Тр. СПИИРАН. – 2017. – Вып. 1. – С. 5-31.
8. *Mittal S., Vetter J.* A survey of CPU-GPU heterogeneous computing techniques // ACM Computing Surveys. – 2015. – Vol. 47. – Art. 69.
9. *Waidyasooriya H.M., Hariyama M., Uchiyama K.* Design of FPGA-Based Computing Systems with OpenCL. – Cham: Springer, 2018. – 126 p.
10. *Tessier R., Pocek K., DeHon A.* Reconfigurable Computing Architectures // Proceedings of the IEEE. – 2015. – Vol. 103, No. 3. – P. 332-354.
11. *Левин И.И., Дордопуло А.И.* К вопросу об автоматическом создании параллельных прикладных программ для реконфигурируемых вычислительных систем // Вычислительные технологии. – 2020. – Т. 25, № 1. – С. 66-81.
12. *Каляев А.В., Левин И.И.* Модульно-наращиваемые многопроцессорные системы со структурно-процедурной организацией вычислений. – М.: Янус-К, 2003. – 380 с.
13. *Левин И.И., Дордопуло А.И., Гудков В.А. и др.* Средства программирования реконфигурируемых и гибридных вычислительных систем на основе ПЛИС // XIII Междунар. конф. «Параллельные вычислительные технологии» (ПаВТ-2019), короткие статьи и описания плакатов. – Челябинск: Изд. центр ЮУрГУ, 2019. – С. 299-312.
14. *Писаренко И.В., Алексеев К.Н., Мельников А.К.* Ресурсонезависимое представление сортирующих сетей на языке программирования Set@l // Вестник компьютерных и информационных технологий. – 2019. – № 11. – С. 53-60.
15. *Levin I.I., Dordopulo A.I., Pisarenko I.V., Melnikov A.K.* Aspect-Oriented Set@l Language for Architecture-Independent Programming of High-Performance Computer Systems // Communications in Computer and Information Science. – 2019. – Vol. 1129. – P. 517-528.

16. *Левин И.И., Дордопуло А.И., Писаренко И.В., Мельников А.К.* Язык архитектурно-независимого программирования вычислительных систем Set@l // Вестник компьютерных и информационных технологий. – 2019. – № 3. – С. 48-56.
17. *Кнут Д.Э.* Искусство программирования. Т. 1. Основные алгоритмы. – 3-е изд. – М.: ООО «И.Д. Вильямс», 2017. – 720 с.
18. *Дасгупта С., Пападимитриу Х., Вазирани У.* Алгоритмы: пер. с англ. / под ред. А. Шеня. – М.: МЦНМО, 2014. – 320 с.
19. *Коваленко А.Г., Левин И.И., Мельников А.К.* Автоматизация построения параллельно-конвейерных программ для реконфигурируемых вычислительных систем // Вестник компьютерных и информационных технологий. – 2013. – № 5. – С. 50-56.
20. *Levin I.I., Dordopulo A.I., Pisarenko I.V., Melnikov A.K.* Objects of Alternative Set Theory in Set@l Programming Language // Lecture Notes in Computer Science. – 2019. – Vol. 11657. – P. 18-31.

REFERENCES

1. *Knut D.E.* Iskusstvo programmirovaniya [The Art of Computer Programming]. Vol. 4, A. *Kombinatornye algoritmy* [Combinatorial Algorithms]. Part 1: transl. from engl. Moacow: OOO «I. D. Vil'yams», 2013, 960 p.
2. *Novikov F.* Diskretnaya matematika [Discrete Mathematics]. 3rd ed. Saint Petersburg: Piter, 2019, 496 p.
3. *Karepova E.D.* Osnovy mnogopotochnogo i parallel'nogo programmirovaniya [Fundamentals of Multithreaded and Parallel Programming]. Krasnoyarsk: Sib. feder. un-t, 2016, 356 p.
4. Zadacha summirovaniya elementov massiva [Problem of Array Elements' Summation], *Laboratorii Parallel'nykh informatsionnykh tekhnologiy NIVTS MGU* [The Laboratory of Parallel Information Technologies of the Research Computing Center of the Moscow State University]. Available at: https://parallel.ru/fpga/Summ2 (accessed 27 April 2020).
5. *Starchenko A.V., Bertsun V.N.* Metody parallel'nykh vychisleniy [Methods of Parallel Computing]. Tomsk: Izd-vo Tom. un-ta, 2013, 223 p.
6. *Efimov S.S.* Obzor metodov rasparallelivaniya algoritmov resheniya nekotorykh zadach vychislitel'noy diskretnoy matematiki [Review of Parallelizing Methods for Algorithms Aimed at Solution of Certain Problems of Computational Discrete Mathematics], *Matematicheskie struktury i modelirovanie* [Mathematical Structures and Modeling], 2007, Issue 17, pp. 72-93.
7. *Kalyaev I.A., Levin I.I., Semernikov E.A., Shmoylov V.I.* Razvitie otechestvennykh mnogokristall'nykh rekonfiguriruemykh vychislitel'nykh sistem: ot vozdushnogo k zhidkostnomu okhlazhdeniyu [Evolution Domestic of Multichip Reconfigurable Computer Systems: from Air to Liquid Cooling: From Air to Liquid Cooling], *Tr. SPIIRAN* [SPIIRAS Proceedings], 2017, Issue 1, pp. 5-31.
8. *Mittal S., Vetter J.* A survey of CPU-GPU heterogeneous computing techniques, *ACM Computing Surveys*, 2015, Vol. 47, Art. 69.
9. *Waidyasooriya H.M., Hariyama M., Uchiyama K.* Design of FPGA-Based Computing Systems with OpenCL. Cham: Springer, 2018, 126 p.
10. *Tessier R., Pocek K., DeHon A.* Reconfigurable Computing Architectures, *Proceedings of the IEEE*, 2015, Vol. 103, No. 3, pp. 332-354.
11. *Levin I.I., Dordopulo A.I.* K voprosu ob avtomaticheskom sozdanii parallel'nykh prikladnykh programm dlya rekonfiguriruemykh vychislitel'nykh sistem [On the problem of automatic development of parallel applications for reconfigurable computer systems], *Vychislitel'nye tekhnologii* [Computational Technologies], 2020, Vol. 25, No. 1, pp. 66-81.
12. *Kalyaev A.V., Levin I.I.* Modul'no-narashchivaemye mnogoprotsessornye sistemy so strukturno-protsedurnoy organizatsiey vychisleniy [Modular-Expandable Multiprocessor Systems with Structural and Procedural Organization of Calculations]. Moscow: YAnus-K, 2003, 380 p.
13. *Levin I.I., Dordopulo A.I., Gudkov V.A. i dr.* Sredstva programmirovaniya rekonfiguriruemykh i gibridnykh vychislitel'nykh sistem na osnove PLIS [Tools for Programming of Reconfigurable and Hybrid Computer Systems Based on FPGAs], *XIII Mezhdunar. konf. «Parallel'nye vychislitel'nye tekhnologii» (PaVT-2019), korotkie stat'i i opisaniya plakatov* [XIII International Conference «Parallel computational technologies» (PCT'2019), short papers and poster descriptions]. Chelyabinsk: Izd. tsentr YuUrGU, 2019, pp. 299-312.

14. *Pisarenko I.V., Alekseev K.N., Mel'nikov A.K.* Resursonezavisimoe predstavlenie sortiruyushchikh setey na yazyke programmirovaniya Set@l [Resource-Independent Representation of Sorting Networks in Set@l Programming Language], *Vestnik komp'yuternykh i informatsionnykh tekhnologiy* [Herald of Computer and Information Technologies], 2019, No. 11, pp. 53-60.
15. *Levin I.I., Dordopulo A.I., Pisarenko I.V., Melnikov A.K.* Aspect-Oriented Set@l Language for Architecture-Independent Programming of High-Performance Computer Systems, *Communications in Computer and Information Science*, 2019, Vol. 1129, pp. 517-528.
16. *Levin I.I., Dordopulo A.I., Pisarenko I.V., Mel'nikov A.K.* Yazyk arkhitekturno-nezavisimogo programmirovaniya vychislitel'nykh sistem Set@l [Architecture-Independent Set@l Programming Language for Computer Systems], *Vestnik komp'yuternykh i informatsionnykh tekhnologiy* [Herald of Computer and Information Technologies], 2019, No. 3, pp. 48-56.
17. *Knut D.E.* Iskusstvo programmirovaniya. T. 1. Osnovnye algoritmy [The Art of Computer Programming. Vol. 1. Fundamental Algorithms]. 3rd ed. Moscow: OOO «I.D. Vil'yams», 2017, 720 p.
18. *Dasgupta S., Papadimitriu Kh., Vazirani U.* Algoritmy [Algorithms]: transl. from engl., ed. by A. Shenya. Moscow: MTSNMO, 2014, 320 p.
19. *Kovalenko A.G., Levin I.I., Mel'nikov A.K.* Avtomatizatsiya postroeniya parallel'no-konveyernykh programm dlya rekonfiguriruemykh vychislitel'nykh sistem [Automatization of parallel-pipeline program development for reconfigurable computer systems], *Vestnik komp'yuternykh i informatsionnykh tekhnologiy* [Herald of Computer and Information Technologies], 2013, No. 5, pp. 50-56.
20. *Levin I.I., Dordopulo A.I., Pisarenko I.V., Melnikov A.K.* Objects of Alternative Set Theory in Set@l Programming Language, *Lecture Notes in Computer Science*, 2019, Vol. 11657, pp. 18-31.

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

Левин Илья Израилевич – Южный федеральный университет; e-mail: iilevin@sfedu.ru; 347928, г. Таганрог, ул. Петровская, 15, кв. 143; тел.: 88634612111; и.о. зав. кафедрой интеллектуальных и многопроцессорных систем; д.т.н., профессор.

Писаренко Иван Вадимович – e-mail: ivan123tgn@yandex.ru; 347942, г. Таганрог, ул. Нижняя линия, 8; тел.: 89185978535; аспирант.

Михайлов Денис Васильевич – e-mail: mixailow.den@gmail.com; 347900, г. Таганрог, ул. Фрунзе, 61, кв. 11; тел.: 89287502869; аспирант.

Дордопуло Алексей Игоревич – Общество с ограниченной ответственностью «НИЦ супер-ЭВМ и нейрокомпьютеров». e-mail: dordopulo@superevm.ru; 347902, г. Таганрог, 9-й переулок, 44; тел.: 88634477407; начальник отдела математико-алгоритмического обеспечения; к.т.н.

Levin Ilya Izrailevich – Southern Federal University. e-mail: iilevin@sfedu.ru; 15, Petrovskaya street, ap. 143, Taganrog, 347928, Russia; phone: +78634612111; head of the department of intellectual and multiprocessor systems; dr. of eng. sc.; professor.

Pisarenko Ivan Vadimovich – e-mail: ivan123tgn@yandex.ru; 8, Nizhnjaja Linija street, Taganrog, 347942, Russia; phone: +79185978535; postgraduate student.

Mikhailov Denis Vasilevich – e-mail: mixailow.den@gmail.com; 61, Frunze stryye, ap. 11, Taganrog, 347900, Russia; phone: +79287502869; postgraduate student.

Dordopulo Alexey Igorevich – "Supercomputers and Neurocomputers Research Center" Co. Ltd.; e-mail: dordopulo@superevm.ru; 44, 9th lane, Taganrog, 347902, Russia; phone: +78634477407; head of the division of mathematic and algorithmic support; cand. of eng. sc.