МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ МАШИН, КОМПЛЕКСОВ И КОМПЬЮТЕРНЫХ СЕТЕЙ

DOI: 10.18698/0236-3933-2016-1-112-128

УДК 004.2:004.31

МЕТОДИКА ДЕКОМПОЗИЦИИ ИНФОРМАЦИОННОГО ГРАФА ПРОГРАММЫ ДЛЯ ОРГАНИЗАЦИИ ПАРАЛЛЕЛЬНОЙ ОБРАБОТКИ ДАННЫХ НА ЭВМ МКОД

В.Э. Подольский[1,2], **А.Ю. Попов**[1]

[1]МГТУ им. Н.Э. Баумана, Москва, Российская Федерация
e-mail: alexpopov@bmstu.ru

[2]ИБС Софт, Москва, Российская Федерация
e-mail: v.e.podolskiy@gmail.com

В МГТУ им. Н.Э. Баумана разрабатывается принципиально новая вычислительная система со многими потоками команд и одним потоком данных (МКОД), в составе которой имеются аппаратные средства для ускорения алгоритмов дискретной оптимизации. В ходе проведенных исследований полученной системы стало очевидно, что для ее эффективного внедрения необходимо модифицировать существующие алгоритмы и адаптировать их под архитектурные особенности МКОД-системы. Однако модификация каждого последовательного алгоритма к требуемому параллельному виду является трудоемким процессом. Поэтому актуальна разработка формальных подходов для автоматизированного преобразования алгоритмов. Предложен способ представления алгоритма МКОД в виде графовой модели, показано решение задачи декомпозиции информационного графа последовательной программы на графы арифметико-логической обработки и обработки структур данных, приведен пример представления информационного графа алгоритма на языке R.

***Ключевые слова*:** МКОД-система, процессор обработки структур, информационный граф алгоритма, декомпозиция графа.

GRAPH DECOMPOSITION FOR PARALLEL DATA PROCESSING ON MISD COMPUTER

V.E. Podolsky[1,2], **A.Yu. Popov**[1]

[1]Bauman Moscow State Technical University, Moscow, Russian Federation
e-mail: alexpopov@bmstu.ru

[2]IBS Soft, Moscow, Russian Federation
e-mail: v.e.podolskiy@gmail.com

A conceptually new computing system dealing with multiple instruction stream and single data stream (MISD) is being developed at Bauman Moscow State Technical University. The system contains the hardware to accelerate discrete optimization algorithms. The MISD studies resulted in the conclusion that its effective implementation demands to modify the occurring algorithms and adapt them to the

architectural MISD features. However, modification of each sequential algorithm to a required parallel representation is a time-consuming process. So, the development of formal approaches for the automated algorithm transformation is urgent. In the paper the way of MISD algorithm representation as a graph model is proposed. Also, the task of graph decomposition of a sequential program into arithmetic-logic processing graphs and data structure processing graphs has been solved. The representation of the algorithm graph in the R programming language is given.

***Keywords*:** multiple instruction stream single data stream computer system (MISD-system), structure handling processor, algorithm graph, graph decomposition.

Введение. Американский ученый М. Флинн в 1966 г. предложил классификацию архитектур ЭВМ по признакам наличия параллелизма в потоках команд и данных: с одним потоком команд и одним потоком данных (ОКОД); с одним потоком команд и многими потоками данных (ОКМД); со многими потоками команд и одним потоком данных (МКОД); со многими потоками команд и многими потоками данных (МКМД). Для трех классов (ОКОД, ОКМД, МКМД) создано немало действующих образцов ЭВМ и систем. Долгое время класс ЭВМ МКОД не был представлен ни одной ЭВМ [1]. Только в 2008 г. была запатентована первая в своем роде архитектура ЭВМ класса МКОД [2]. В настоящее время ЭВМ МКОД разрабатывается в МГТУ им. Н.Э. Баумана.

Принципы обработки одного потока данных различными потоками команд в предложенной архитектуре ЭВМ МКОД реализовывались за счет наличия двух процессоров [3]: центрального процессора (ЦП), выполняющего арифметико-логические операции, и процессора обработки структур (СП), выполняющего операции над структурами данных. Вследствие дихотомической природы данных удалось реализовать параллельную обработку структурной и информационной составляющих данных на соответствующих процессорах. Архитектура предложенной системы представлена на рис. 1.

Ввиду аппаратной поддержки операций над структурами стало возможным описать обработку структурированных данных с помощью основных операций теории множеств. На аппаратном уровне СП поддерживаются следующие команды: *AND* (пересечение множеств); *OR* (объединение множеств); *SEARCH* (доступ к элементу множества); *INSERT* (добавление элемента множества); *DELETE* (удаление элемента множества); *DELSTR* (удаление структуры); *MAX* (поиск максимального элемента множества); *MIN* (поиск минимального элемента множества); *POWER* (определение числа элементов множества); команды получения срезов множества (меньше *LS*, меньше или равно *LSEQ*, больше *GR*, больше или равно *GREQ*); *NOT* (разность множеств); *NEXT* (переход к следующему элементу множества); *JT* (операция перехода по тегу).

Основная проблема архитектуры МКОД — необходимость адаптации существующих последовательных и параллельных алгоритмов с

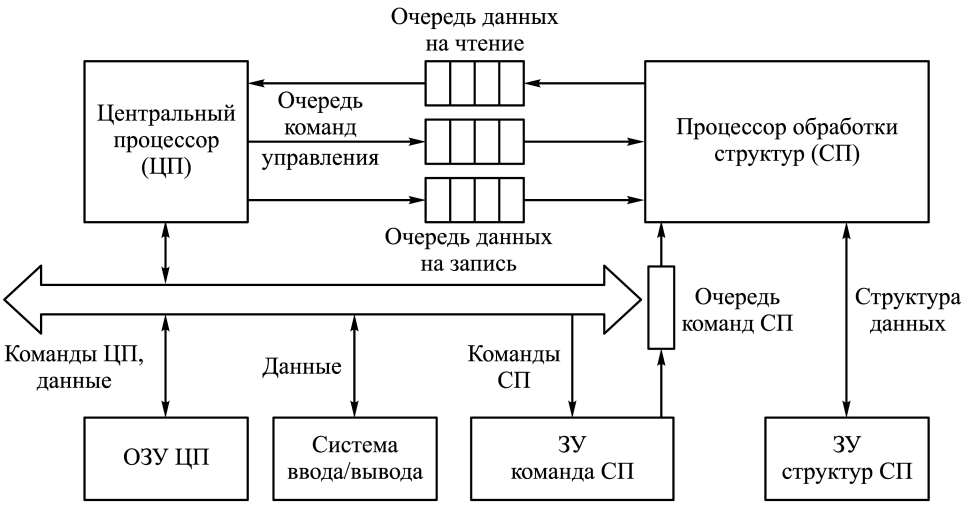


Рис. 1. Архитектура ЭВМ МКОД

учетом разделения потока команд на поток команд обработки структур и поток команд обработки информационной составляющей данных. Одним из ключевых факторов, сдерживающих развитие архитектуры, как было определено при ручной адаптации алгоритмов Дейкстры [4], Форда – Фалкерсона [5], Беллмана – Форда и Ли [6], оказалась высокая трудоемкость модификации последовательных алгоритмов для параллельной обработки данных на ЭВМ МКОД. Поэтому в работе [6] было предложено использовать аппарат теории графов для разработки алгоритмов автоматической подготовки программ к выполнению на ЭВМ МКОД. В настоящей работе предложен базовый алгоритм автоматической подготовки программ к выполнению на ЭВМ МКОД, а также приведены примеры кода, реализующего предложенный алгоритм. Впервые проведена формализация задачи разделения кода последовательной программы на две части, которые могут быть параллельно обработаны ЭВМ МКОД, предложена методика, осуществляющая разделение представления кода последовательной программы в виде двух графов, соответствующих последовательности команд обработки данных на ЦП ЭВМ МКОД и последовательности команд обработки структур данных на СП ЭВМ МКОД.

Формальная постановка задачи декомпозиции информационного графа программы. Любая последовательная программа (последовательный алгоритм) может быть представлена с помощью особой интегральной модели, объединяющей взвешенный двудольный ориентированный граф $G_{\text{о.д.}}$ и управляющий граф $G_{\text{у}}$ [7–9]. Каждая инструкция и каждая единица данных представлены в графе отдельной вершиной, причем вершины инструкций связаны непосредственно, тогда как вершины данных связаны опосредованно, через вершины инструкций.

Интегральная модель формально определяется следующим образом:

$$G_A\left(\{X,Y\},F_1X,F_1^{-1}X,F_2X,F_2^{-1}X,F_3Y,F_3^{-1}Y\right)=G_{\text{у}}\cup G_{\text{о.д}},$$

где $X=\{x_1,x_2,\ldots,x_n\}$ — множество вершин графа G_A, соответствующих инструкциям программы; $Y=\{y_1,y_2,\ldots,y_3\}$ — множество вершин графа G_A, соответствующих элементам данных, $X\cap Y=\emptyset$; F_1X — множество вершин-образов вершин потока управления, $F_1X\subset X$; $F_1^{-1}X$ — множество вершин-прообразов вершин потока управления, $F_1^{-1}X\subset X$; F_2X — множество вершин-образов вершин потока управления во множестве вершин, соответствующих обрабатываемым данным, $F_2X\subset Y$; $F_2^{-1}X$ — множество вершин-прообразов вершин потока управления во множестве вершин, соответствующих обрабатываемым данным, $F_2^{-1}X\subset Y$; F_3Y — множество вершин-образов вершин данных во множестве вершин потока управления, $F_3Y\subset X$; $F_3^{-1}Y$ — множество вершин-прообразов вершин данных во множестве вершин потока управления, т.е. $F_3^{-1}Y\subset X$.

Пример информационного графа, построенного в соответствии с описанной моделью, приведен на рис. 2. Поскольку интегральная модель алгоритма не делает различий между примитивными типами данных и структурами, необходимо ее детализировать для применения к решению задачи преобразования последовательных программ в вид, обрабатываемый ЭВМ МКОД.

Обозначим множество всех инструкций (операторов) исходной последовательной программы (алгоритма) O, множество данных, принадлежащих области определения D. При этом примем, что $O=O^*\cup O^{**}$, $O^*\cap O^{**}=\emptyset$, где O^* — множество операторов обработки данных; O^{**} — множество операторов вычисления условий.

Детализируем множество операторов обработки данных

$$O^*=O_S^*\cup O_I^*, \tag{1}$$

где O_S^* — множество операторов обработки структур данных; O_I^* — множество операторов обработки данных примитивных типов, в том числе отдельных элементов структур. Подобная детализация (1) позволяет разделить все операторы на две пересекающиеся категории:

1) операторы, которые возможно выполнить в СП ($O_{\text{СП}}=O_S^*$);
2) операторы, которые возможно выполнить в ЦП ($O_{\text{ЦП}}=O_I^*\cup O_S^*=$ $=O^*$).

Одна из особенностей функционирования ЭВМ МКОД – действия исходного алгоритма могут выполняться как в ЦП, так и в СП, т.е. отсутствует строгая определенность при назначении обрабатывающего процессора. Центральный процессор представляет собой универсальное устройство, способное выполнить любое действие (возможно, за большее время), в то время как СП может выполнять только перечи-

сленные ранее команды обработки структур (возможно, за меньшее время). Поэтому, $O_{\text{СП}} \subset O_{\text{ЦП}}$.

Основная цель проекта ЭВМ МКОД — разработка аппаратных механизмов, повышающих эффективность вычислительной системы в целом при обработке структур данных, множеств, графов. Полагая такой результат возможным, примем, что СП затрачивает на обработку действий меньше ресурсов (времени, рассеиваемой мощности, удельной стоимости) по сравнению с ЦП. Тогда можно выделить ключевую задачу подготовки программы для ЭВМ МКОД как задачу минимизации мощности множества операторов, которые действительно выполняются в ЦП, за счет выполнения как можно большего числа операторов в СП. Обозначим множество возможных значений операторов обработки структур $O^*_{S\text{возм}}$ (его компоненты — базовые операции теории множеств, приведенные выше).

Для формализации задачи декомпозиции информационного графа алгоритма введем несколько дополнительных обозначений: $O^{\text{действ}}_{\text{ЦП}}$, $O^{\text{действ}}_{\text{СП}}$ — множества операторов, которые действительно выполняются ЦП и СП после осуществления декомпозиции; $O_{\leftrightarrow} = O_{\rightarrow} \cup O_{\leftarrow}$ — множество операторов обмена данными между ЦП и СП ($O_{\rightarrow}$ — множество операторов пересылки данных; $O_{\leftarrow}$ — множество операторов получения данных). Следует отметить, что $O_{\leftrightarrow} \cap O = \emptyset$, поэтому введем понятие полного множества операторов ЭВМ МКОД: $O_{\text{МКОД}} = O \cup O_{\leftrightarrow}$.

Аналогично выделим виды обрабатываемых программой данных. Множество обрабатываемых данных D представлено подмножествами структур данных ($D_S \subset D$) и данных примитивного типа ($D_P \subset D$). Таким образом, получим $D = D_S \cup D_P$. Поскольку структуры данных состоят из элементов данных (примитивного типа) и отношений между ними, каждая структура данных может быть представлена в виде $d^S_i = (DE_i, R_i)$, где DE_i — множество элементов данных примитивного типа i-й структуры данных; R_i — множество отношений между отдельными элементами DE_i i-й структуры данных, $d^S_i \in D_S$, $DE_i \in D_P$.

Опишем детализированную модель в терминах теории графов для получения формализованного представления результата декомпозиции информационного графа программы. Введем дополнительные обозначения: X_S — множество вершин обработки структур данных, $O^{\text{действ}}_{\text{СП}} \leftrightarrow X_S$, $X_S \subset X$; X_I — множество вершин обработки данных примитивных типов, $O^*_I \leftrightarrow X_I$, $X_I \subset X$; X_C — множество вершин вычисления условий, $O^{**} \leftrightarrow X_C$, $X_C \subset X$; $X_{\text{МКОД}}$ — полное множество вершин информационного графа программы для ЭВМ МКОД, представляющее собой отображение $O \cup O_{\leftrightarrow} \leftrightarrow X_{\text{МКОД}}$, $X \subset X_{\text{МКОД}}$; $X_{\leftrightarrow}$ — множество операторов обмена данными между ЦП и СП, $O_{\leftrightarrow} \leftrightarrow X_{\leftrightarrow}$, $X_{\leftrightarrow} \subset X_{\text{МКОД}}$; $X_{IC\leftrightarrow}$ — множество вершин обработки элементов данных, данных примитивных типов и вычисления условий, а также

передачи данных ЦП СП, $X_{IC\leftrightarrow} = X_I \cup X_C \cup X_{\leftrightarrow}$; $X_{S\leftrightarrow}$ — множество вершин обработки структур данных, а также передачи данных ЦП СП, $X_{S\leftrightarrow} = X_S \cup X_{\leftrightarrow}$; Y_S — множество вершин структур данных, $D_S \leftrightarrow Y_S$, $Y_S \subset Y$; Y_P — множество вершин данных примитивного типа $D_P \leftrightarrow Y_P$, $Y_P \subset Y$.

Для полного формального описания декомпозированного информационного графа программы также должны быть формально определены отношения между элементами множеств вершин: $F_1X_{IC\leftrightarrow}$ — множество вершин-образов вершин потока управления в ЦП, $F_1X_{IC\leftrightarrow} \subset X_{IC\leftrightarrow}$; $F_1^{-1}X_{IC\leftrightarrow}$ — множество вершин-прообразов вершин потока управления в ЦП, $F_1^{-1}X_{IC\leftrightarrow} \subset X_{IC\leftrightarrow}$; $F_1X_{S\leftrightarrow}$ — множество вершин-образов вершин потока управления в СП, $F_1X_{S\leftrightarrow} \subset X_{S\leftrightarrow}$; $F_1^{-1}X_{S\leftrightarrow}$ — множество вершин-прообразов вершин потока управления в СП, $F_1^{-1}X_{S\leftrightarrow} \subset X_{S\leftrightarrow}$; $F_2X_{IC\leftrightarrow}$ — множество вершин-образов вершин потока управления во множестве вершин, соответствующих обрабатываемым в ЦП данным, $F_2X_{IC\leftrightarrow} \subset Y_P$ (если $d_1 \leftrightarrow y_1 \in Y_P$ является результатом работы инструкции $x_1 \in X_{IC\leftrightarrow}$, то $F_2x_1 = y_1$); $F_2^{-1}X_{IC\leftrightarrow}$ — множество вершин-прообразов вершин потока управления во множестве вершин, соответствующих обрабатываемым в ЦП данным, $F_2^{-1}X_{IC\leftrightarrow} \subset Y_P$ (если $d_1 \leftrightarrow y_1 \in Y_P$ является исходным данным для инструкции $x_1 \in X_{IC\leftrightarrow}$, то $F_2^{-1}x_1 = y_1$); $F_2X_{S\leftrightarrow}$ — множество вершин-образов вершин потока управления во множестве вершин, соответствующих обрабатываемым в СП данным, $F_2X_{S\leftrightarrow} \subset Y_S$ (если $d_1 \leftrightarrow y_1 \in Y_S$ является результатом работы инструкции $x_1 \in X_{S\leftrightarrow}$, то $F_2x_1 = y_1$); $F_2^{-1}X_{S\leftrightarrow}$ — множество вершин-прообразов вершин потока управления во множестве вершин, соответствующих обрабатываемым СП данным, $F_2^{-1}X_{S\leftrightarrow} \subset Y_S$ (если $d_1 \leftrightarrow y_1 \in Y_S$ является исходным данным для инструкции $x_1 \in X_{S\leftrightarrow}$, то $F_2^{-1}x_1 = y_1$); F_3Y_P — множество вершин-образов вершин данных, обрабатываемых в ЦП, во множестве вершин потока управления, выполняемого ЦП, $F_3Y_P \subset X_{IC\leftrightarrow}$ (если $d_1 \leftrightarrow y_1 \in Y_P$ является исходным данным для инструкции $x_1 \in X_{IC\leftrightarrow}$, то $F_3y_1 = x_1$; $F_3^{-1}Y_P$ — множество вершин-прообразов вершин данных, обрабатываемых в ЦП, во множестве вершин потока управления, выполняемого ЦП, $F_3^{-1}Y_P \subset X_{IC\leftrightarrow}$ (если $d_1 \leftrightarrow y_1 \in Y_P$ является результатом обработки для инструкции $x_1 \in X_{IC\leftrightarrow}$, то $F_3^{-1}y_1 = x_1$); F_3Y_S — множество вершин-образов вершин данных, обрабатываемых в СП, во множестве вершин потока управления, выполняемого СП, $F_3Y_S \subset X_{S\leftrightarrow}$ (если $d_1 \leftrightarrow y_1 \in Y_S$ является исходным данным для инструкции $x_1 \in X_{S\leftrightarrow}$, то $F_3y_1 = x_1$); $F_3^{-1}Y_S$ — множество вершин-прообразов вершин данных, обрабатываемых в СП, во множестве вершин потока управления, выполняемого СП, $F_3^{-1}Y_S \subset X_{S\leftrightarrow}$ (если $d_1 \leftrightarrow y_1 \in Y_S$ является результатом обработки для инструкции $x_1 \in X_{S\leftrightarrow}$, то $F_3^{-1}y_1 = x_1$).

В результате декомпозированный информационный граф программы представляет собой систему из двух графов

$$Dec\left(G_A\right)=\begin{cases} G_{AI}\left(\left\{X_{IC\leftrightarrow},Y_P\right\},F_1X_{IC\leftrightarrow},F_1^{-1}X_{IC\leftrightarrow},\right. \\ \qquad\qquad \left. F_2X_{IC\leftrightarrow},F_2^{-1}X_{IC\leftrightarrow},F_3Y_P,F_3^{-1}Y_P\right); \\ G_{AS}\left(\left\{X_{S\leftrightarrow},Y_S\right\},F_1X_{S\leftrightarrow},F_1^{-1}X_{S\leftrightarrow},F_2X_{S\leftrightarrow},\right. \\ \qquad\qquad \left. F_2^{-1}X_{S\leftrightarrow},F_3Y_S,F_3^{-1}Y_S\right). \end{cases} \quad (2)$$

Здесь G_{AI} — информационный граф алгоритма арифметико-логической обработки данных примитивного типа; G_{AS} — информационный граф алгоритма обработки структур данных.

Декомпозиция информационного графа программы, представленного на рис. 2, в соответствии с введенными выше понятиями приведена на рис. 3.

Для получения декомпозированного информационного графа программы $Dec\left(G_A\right)$, соответствующего программе для ЭВМ МКОД, необходимо определить методику преобразования графа G_A в пару графов $\left(G_{AS},G_{AI}\right)$. Таким образом, ключевая задача заключается в поиске некоторого преобразования или совокупности преобразований Par вида

$$Par:G_A\rightarrow\left(G_{AS},G_{AI}\right). \quad (3)$$

Преобразование (или совокупность преобразований) графа Par (3) позволяет за конечное число шагов получить из информационного графа последовательного алгоритма G_A два графа G_{AS} и G_{AI} таких, что операторы в вершинах графа G_{AS} будут выполняться в СП, а операторы в вершинах графа G_{AI} — в ЦП.

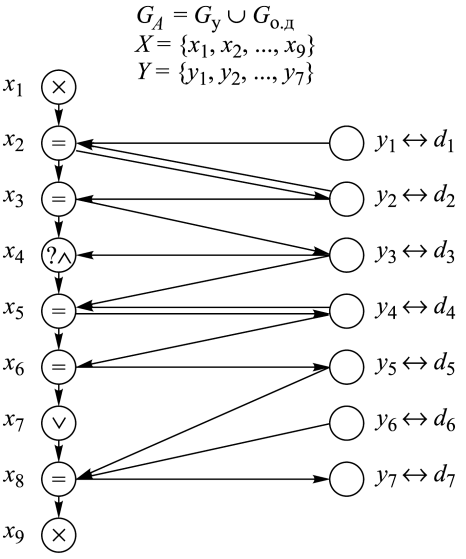


Рис. 2. Пример информационного графа алгоритма:

⊗ — терминальные блоки (x_1 — начало, x_9 — конец); = — обработка данных; ?∧ — объединенный блок вычисления условий и ветвления; ∧ — ветвление потоков управления; ∨ — слияние потоков управления

Декомпозиция информационного графа программы и критерии получения оптимальной декомпозиции информационного графа алгоритма. Основой декомпозиции информационного графа программы должно стать преобразование, выполняющее удаление вершин-операторов X_S, которые отвечают за обработку структур данных, из исходного информационного графа алгоритма.

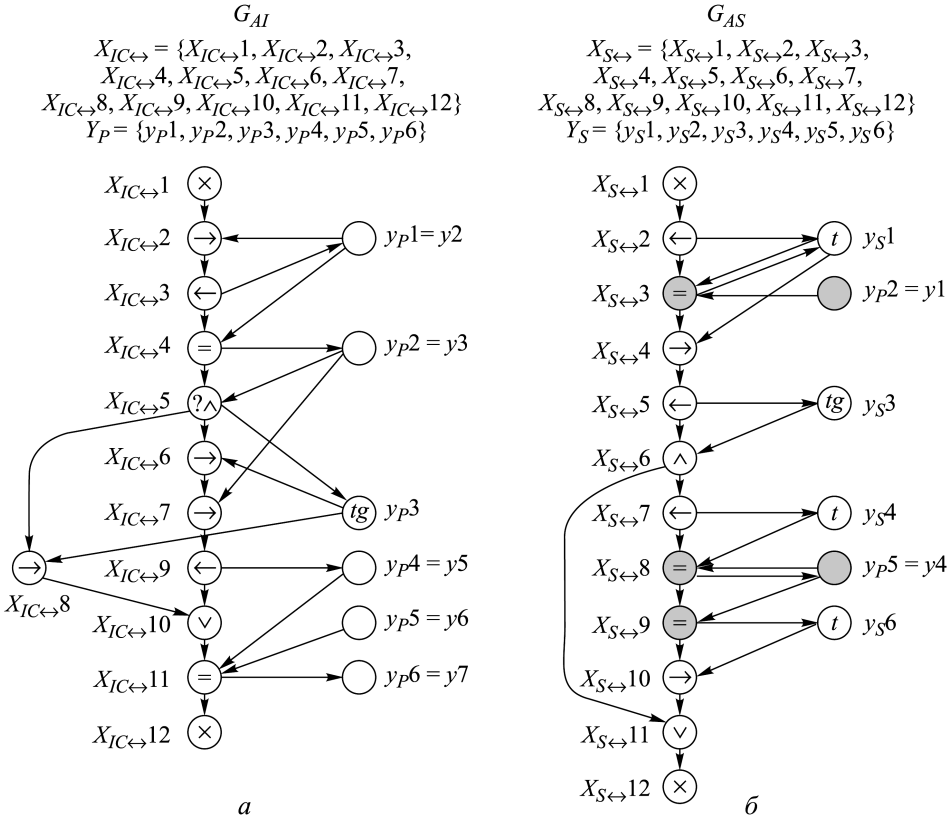


Рис. 3. Декомпозиция информационного графа программы на граф алгоритма арифметико-логической обработки данных примитивного типа G_{AI} (*а*) и информационный граф алгоритма обработки структур данных G_{AS} (*б*):
⊗ — передача данных другому процессу; ⊜ — получение данных от другого процесса; ⊜ — данные примитивного типа; ?∧ — структуры данных; ∧ — тег (примитивный тип); ∨ — временны́е данные примитивного типа (СП)

По аналогии также должны быть удалены все вершины графа Y_S, представляющие собой отображение структур данных на множество вершин. Определим критерии, позволяющие отличить указанные типы вершин графа от остальных.

Критерий 1. Вершина информационного графа алгоритма $y_i \in Y$ является отображением структуры данных d_i на множество вершин данных Y_S тогда и только тогда, когда $d_i = (DE_i, R_i)$, причем $|DE_i| > 1, |R_i| > 0$, или более формально: $y_i \in Y_S : y_i \leftrightarrow d_i$, $d_i = (DE_i, R_i), |DE_i| > 1, |R_i| > 0$.

Критерий 2. Вершина информационного графа алгоритма $x_i \in X$ принадлежит множеству вершин-операторов X_S, отвечающих за обработку структур данных, тогда и только тогда, когда хотя бы одна из смежных ей вершин принадлежит множеству вершин Y_S, т.е. на вход оператора, соответствующего x_i, должна подаваться хотя бы од-

на структура данных, или

$$x_i \in X_S : \exists y_j | y_j | Y_S, F_2^{-1} x_i = y_j. \quad (4)$$

Не нарушая общности дальнейших выводов, предположим, что в примере графа, приведенного на рис. 2, $Y_S = \{y_1, y_4\}$. Согласно (4), в критерии 2 получим $X_S = \{x_2, x_5, x_6\}$, поскольку $F_2^{-1} x_2 = \{y_1, y_2\}, y_1 \in Y_S$; $F_2^{-1} x_5 = \{y_3, y_4\}, y_4 \in Y_S$; $F_2^{-1} x_6 = \{y_4\}, y_4 \in Y_S$. Необходимо на приведенных рисунках визуально отличать вершины информационного графа программы, принадлежащие множествам X_S и Y_S, поэтому они закрашены серым цветом. Поскольку ЦП проводит обработку структуры данных, возможны различные конфигурации множества X_S. Состав множества X_S должен определяться исходя из баланса уменьшения времени обработки от включения очередной вершины в множество X_S и издержек по времени, возникающих при передаче данных СП ЦП. Проблема выбора наиболее оптимальной конфигурации X_S будет подробно рассмотрена в будущих работах.

Последовательность действий в рамках методики декомпозиции информационного графа программы для ЭВМ МКОД представлена ниже.

Декомпозиция информационного графа программы для ЭВМ МКОД. Рассмотрим два этапа декомпозиции: подготовительный этап; этапы преобразования графов $G_A^{(1)}$ и $G_A^{(2)}$.

Подготовительный этап. Шаг 1. Для декомпозиции осуществляется дублирование исходного информационного графа программы: $G_A^{(1)} \leftarrow G_A$, $G_A^{(2)} \leftarrow G_A$. Впоследствии в ходе выполнения преобразования граф $G_A^{(1)}$ будет приведен к виду G_{AI}, а граф $G_A^{(2)}$ — к виду G_{AS}.

Преобразования графа $G_A^{(1)}$. Шаг 2. Пусть определено подмножество $Y_S \subset Y$ вершин-структур данных. Тогда для графа $G_A^{(1)}$ подмножество вершин-операторов обработки структур данных есть $X_S = \{x_i\}$, $i = \overline{1 : |X_S|} : F_2^{-1} x_i = Y', Y' \subset Y_S$.

Шаг 3. Для каждой вершины $x_i \in X_S$, если $y_j \in F_2^{-1} x_i$ и $y_j \in Y \backslash Y_S$, $X \leftarrow X \cup \{x_{\rightarrow}\}$, $F_1 x_{i-1} \leftarrow F_1 x_{i-1} \cup \{x_{\rightarrow}\}$, $F_1^{-1} x_i \leftarrow F_1^{-1} x_i \cup \{x_{\rightarrow}\}$ и $F_2^{-1} x_{\rightarrow} \leftarrow y_j$. В этом действии осуществляется вставка в граф $G_A^{(1)}$ вершины-оператора передачи данных примитивного типа СП.

Шаг 4. Для каждой вершины $x_i \in X_S$, если $y_j \in F_2 x_i$ и $y_j \in Y \backslash Y_S$, $X \leftarrow X \cup \{x_{\leftarrow}\}$, $F_1 x_i \leftarrow F_1 x_i \cup \{x_{\leftarrow}\}$, $F_1^{-1} x_{i+1} \leftarrow F_1^{-1} x_{i+1} \cup \{x_{\leftarrow}\}$ и $F_2 x_{\leftarrow} \leftarrow y_j$. В этом действии выполняется вставка в граф $G_A^{(1)}$ вершины-оператора получения результатов работы операторов в СП данных примитивного типа. Вершина получения данных $x_{\leftarrow}$ от СП не может иметь дублей в графе в силу построения, поскольку такая вершина добавляется для каждой вершины $x_i \in X_S$, записывающей некоторое новое значение в вершину $y_j \in Y \backslash Y_S$.

Шаг 5. Для любого $\forall x_i \in X_C$ добавляем новую вершину данных примитивного типа, хранящую информацию о результате вычисления условия в вершине тега x_i: $Y \leftarrow Y \cup \{y_{tg}\}$, $F_2 x_i = y_{tg}$. Добавляем в граф $G_A^{(1)}$ вершины-операторы передачи информации о теге СП: $X \leftarrow X \cup \{x_{\rightarrow}^{\text{истина}}, x_{\rightarrow}^{\text{ложь}}\}$, $F_1 x_i \leftarrow \{x_{\rightarrow}^{\text{истина}}, x_{\rightarrow}^{\text{ложь}}\}$, $F_1^{-1} x_{i+1} = x_{\rightarrow}^{\text{истина}}$, $F_1^{-1} x_k = x_{\rightarrow}^{\text{ложь}}$, где x_k — вершина в множестве X, соответствующая оператору, который выполняется при ложности условия в вершине x_i, либо вершина слияния потоков управления.

Шаг 6. Осуществляется исключение дублирующих передач данных. Для любой пары вершин-операторов обработки структур x_i и x_j таких, что $\exists y_k \in Y : y_k \in F_2^{-1} x_i, y_k \in F_2^{-1} x_j$, $y_k \notin Y_S$ и между ними в графе X не присутствует других вершин-операторов обработки структур, использующих y_k ($x_{i+1}, \ldots, x_{j-1} \notin F_3 y_k$), проверяем наличие вершин, записывающих новое значение в вершину y_k. Если таких вершин нет, то есть $F_3^{-1} y_k \not\subset \{x_{i+1}, \ldots x_{j-1}\}$, тогда передача данных СП для вершины x_j может быть исключена из графа $G_A^{(1)}$ как дублирующая, следовательно, $XX \leftarrow X \backslash x_{\rightarrow}$ для $x_{\rightarrow} : x_j \in F_1 x_{\rightarrow}$. Проводим "сшивание": $F_1 x_{j-2} \leftarrow F_1 x_{j-2} \cup \{x_j\}$, $F_1^{-1} x_j \leftarrow F_1^{-1} x_j \cup \{x_{j-2}\}$.

Шаг 7. Из множества X удаляются все вершины-операторы обработки структур, т.е. $X \leftarrow X \backslash X_S$, если $x_i \in X_S$, то $F_1 x_{i-1} \leftarrow F_1 x_{i-1} \cup x_{i+1}$, $F_1^{-1} x_{i+1} \leftarrow F_1^{-1} x_{i+1} \cup x_{i-1}$.

Шаг 8. Из множества Y удаляются все вершины, у которых одновременно отсутствуют инцидентные им дуги и дуги, которым они инцидентны, т.е. $Y \leftarrow Y \backslash \{y_i\}$: $F_3 y_i = \emptyset, F_3^{-1} y_i = \emptyset$, $i = \overline{1 : |Y|}$.

Шаг 9. Осуществляется перенумерация вершин в множестве X по порядку, так что обозначение каждой вершины приобретает вид $x_{IC \leftrightarrow i}$, а само множество X переходит во множество $X_{IC \leftrightarrow}$.

Шаг 10. Осуществляется перенумерация вершин в множестве Y по порядку, так что обозначение каждой вершины приобретает вид y_{pi}, а само множество Y переходит во множество Y_p.

Шаг 11. Полученный граф является графом арифметико-логической обработки данных примитивных типов $G_{AI} \leftarrow G_A^{(1)}$.

Преобразование графа $G_A^{(2)}$. Шаг 12. Пусть определено подмножество $Y_S \subset Y$ вершин-структур данных. Тогда для графа $G_A^{(2)}$ получим подмножество вершин-операторов обработки структур данных $X_S = \{x_i\}$, $i = \overline{1 : |X_S|}$: $F_2^{-1} x_i = Y', Y' \subset Y_S$.

Шаг 13. Для каждой вершины $x_i \in X_S$, если $y_j \in F_2 x_i$, $y_j \in Y \backslash Y_S$ и $\exists x_k \in F_3 y_j$, $x_k \notin X_S, k > i$, то $X \leftarrow X \cup \{x_{\rightarrow}\}$. При этом $F_1 x_i \leftarrow F_1 x_i \cup \{x_{\rightarrow}\}$, $F_1^{-1} x_{i+1} \leftarrow F_1^{-1} x_{i+1} \cup \{x_{\rightarrow}\}$ и $F_2^{-1} x_{\rightarrow} \leftarrow y_j$. В этом действии осуществляется вставка в граф $G_A^{(2)}$ вершины-оператора передачи данных примитивного типа ЦП. Вершины отправки данных

$x_{\rightarrow}$ в ЦП не могут иметь дублей в графе в силу построения, поскольку такая вершина добавляется для каждой вершины $x_i \in X_S$, записывающей некоторое новое значение в вершину $y_j \in Y \backslash Y_S$.

Шаг 14. Для каждой вершины $x_i \in X_S$, если $y_j \in F_2^{-1}x_i$, $y_j \in Y \backslash Y_S$ и $\exists x_k \in F_3^{-1}y_j$, $x_k \notin X_S, k < i$, то $X \leftarrow X \cup \{x_{\leftarrow}\}$. При этом $F_1 x_{i-1} \leftarrow F_1 x_{i-1} \cup \{x_{\leftarrow}\}$, $F_1^{-1}x_i \leftarrow F_1^{-1}x_i \cup \{x_{\leftarrow}\}$ и $F_2 x_{\leftarrow} \leftarrow y_j$, т.е. в данном действии происходит вставка в граф $G_A^{(2)}$ вершины-оператора получения данных примитивного типа от ЦП.

Шаг 15. У любой вершины $\forall x_i \in X_C$, являющейся объединенным блоком вычисления условия и ветвления, заменяем вершину ветвления потоков управления, если $x_i \in X_{?\Lambda}$, то $x_i \leftarrow x_{\Lambda i}$.

Шаг 16. Для каждой вершины ветвления потоков управления $x_i \in X_\Lambda$ добавляем новую вершину-оператор получения данных от ЦП $x_{\leftarrow}$: $X \leftarrow X \cup \{x_{\leftarrow}\}$, $F_1 x_{i-1} \leftarrow F_1 x_{i-1} \cup \{x_{\leftarrow}\}$, $F_1^{-1}x_i \leftarrow F_1^{-1}x_i \cup \{x_{\leftarrow}\}$. В граф добавляем вершину данных примитивного типа, хранящую информацию о результате вычисления условия ЦП в вершине тега x_i $Y \leftarrow Y \cup \{y_{tg}\}$, $F_2^{-1}x_i = y_{tg}$.

Шаг 17. Проводится замена используемых вершинами-операторами обработки структур данных вершин-данных примитивных типов вершинами временны́х данных примитивного типа, т.е. $\forall y_i \notin Y_S : x_j \in \in F_3 y_i, x_j \in X_S$ заменяем вершину y_i вершиной временны́х данных, сохраняя все дуги: $y_i \leftarrow y_i^t$.

Шаг 18. Осуществляется исключение дублирующих передач данных СП. Для любой пары вершин-операторов обработки структур x_i и x_j, таких, что $\exists y_k \in Y : y_k \in F_2^{-1}x_i, y_k \in F_2^{-1}x_j$, $y_k \notin Y_S$ и между ними в графе не присутствует других вершин-операторов обработки структур, использующих y_k (т.е. $x_{i+1}, \dots x_{j-1} \notin F_3 y_k$), проверяем наличие вершин, записывающих новое значение в вершину y_k. Если таких вершин нет, т.е. $F_3^{-1}y_k \not\subset \{x_{i+1}, \dots x_{j-1}\}$, то передача данных СП для вершины x_j может быть исключена из графа $G_A^{(2)}$ как дублирующаяся, следовательно, $X \leftarrow X \backslash x_{\leftarrow}$ для $x_{\leftarrow} : x_j \in F_1 x_{\leftarrow}$. Проводим "сшивание" $F_1 x_{j-2} \leftarrow F_1 x_{j-2} \cup \{x_j\}$, $F_1^{-1}x_j \leftarrow F_1^{-1}x_j \cup \{x_{j-2}\}$.

Шаг 19. Из множества X удаляются все вершины-операторы, не являющиеся операторами обработки структур, передачи, а также терминальными или операторами ветвления, т.е. $X \leftarrow X \backslash X_p$, если $x_i \in X_p$, то $F_1 x_{i-1} \leftarrow F_1 x_{i-1} \cup x_{i+1}$,$F_1^{-1}x_{i+1} \leftarrow F_1^{-1}x_{i+1} \cup x_{i-1}$.

Шаг 20. Из множества Y удаляются все вершины, у которых отсутствуют инцидентные им дуги и которые не инцидентны никаким дугам: $Y \leftarrow Y \backslash \{y_i\}$: $F_3 y_i = \emptyset, F_3^{-1}y_i = \emptyset$, $i = \overline{1 : |Y|}$.

Шаг 21. Осуществляется перенумерация вершин в множестве X по порядку, так что обозначение каждой вершины приобретает вид $x_{S\leftrightarrow i}$, а само множество X переходит в множество $X_{S\leftrightarrow}$.

Шаг 22. Выполняется перенумерация вершин в множестве Y по порядку, так что обозначение каждой вершины приобретает вид y_{Si}, а само множество Y переходит в множество Y_S.

Шаг 23. Полученный граф является графом обработки структур данных: $G_{AS} \leftarrow G_A^{(2)}$.

Среди характерных свойств полученных графов G_{AS} и G_{AI} можно выделить их структурное сходство с исходным информационным графом последовательного алгоритма, в том числе то, что они также представляют собой объединение графа и ориентированного двудольного графа. Более важное свойство — полное отсутствие у полученных графов общих дуг, что позволяет в дальнейшем восстановить по графам исходный код программы, которая может быть выполнена на ЭВМ МКОД: граф G_{AS} соответствует программе, которая будет выполняться СП, а граф G_{AI} — программе, которая будет выполняться ЦП. Несмотря на эту верхнеуровневую независимость, делающую возможной выполнение программы, описываемой графами, на ЭВМ МКОД между графами и программами для ЦП и СП имеются зависимости по данным, показанные наличием вершин передачи данных $x_{\leftarrow}$ и $x_{\rightarrow}$ в графах. Снижение числа вершин передачи данных может рассматриваться как один из критериев качества получаемого разбиения информационного графа последовательной программы на два графа.

Рассмотрим некоторые ключевые преимущества и недостатки проведенной декомпозиции информационного графа последовательной программы для ЭВМ МКОД (таблица).

Проведенная декомпозиция в полной мере позволяет решить задачу декомпозиции информационного графа последовательной программы для ЭВМ МКОД. Тем не менее декомпозиция обладает недостатками, устранение которых позволит получить более качественное разбиение информационного графа последовательного алгоритма. Возможности улучшения способов декомпозиции информационного графа будут рассмотрены в будущих работах.

Подход к программной реализации методики декомпозиции информационного графа программы для ЭВМ МКОД. Для программной реализации описанной методики декомпозиции информационного графа последовательной программы для ЭВМ МКОД было применено аналитическое описание графов и операций над ними. Основные типы графов и их компоненты, операции над ними на языке теории множеств, а также подходы к визуализации описаны в пакетах “graph” и “Rgraphviz” интерпретируемого языка программирования R, предназначенного для анализа массивов и структур данных [10, 11]. Методика декомпозиции была реализована на языке R и протестирована на примере графа, представленного на рис. 2. В результате тестирования получены части информационного графа алгоритма, показанные

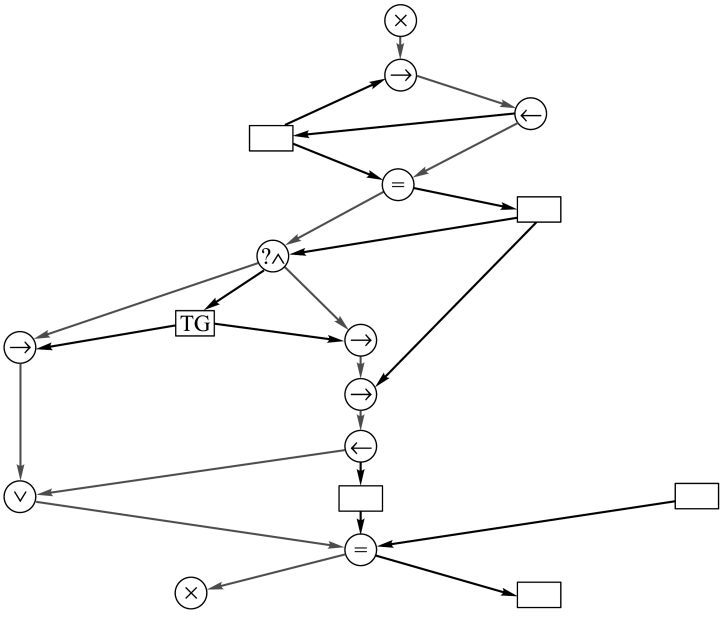
Преимущества и недостатки декомпозиции информационного графа программы (алгоритма) для ЭВМ МКОД

Преимущество	Недостаток	Способ устранения недостатка
Возможность автоматического проведения декомпозиции информационного графа.	Отсутствие единого критерия оценки качества полученного разбиения информационного графа программы.	Использование различных критериев для оценки качества разбиения.
Высокоуровневое представление программы позволяет быстро выявлять и устранять основные зависимости частей декомпозируемого графа, а также обеспечивать наглядное представление результата для последующего анализа.	Сложность обнаружения проблем, связанных с распараллеливанием на уровне команд (связи по данным между итерациями циклов и пр.).	После (или во время) декомпозиции проведение модификации полученных частей программы с помощью существующих средств анализа и устранения зависимостей по данным на низком уровне.
Использование минимального числа разновидностей и вспомогательных вершин в процессе декомпозиции.	Не учитывается вариант частичного выполнения СП операторов обработки структур данных.	Модификация и применение декомпозиции с переносом в граф G_{AS} различных наборов вершин обработки структур данных с последующим выбором наиболее оптимальной декомпозиции согласно некоторому критерию.

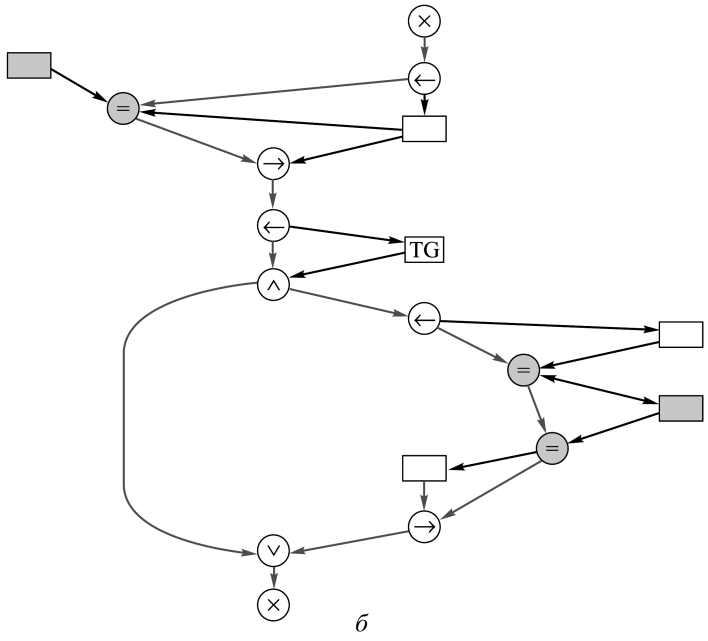
на рис. 4. Графы изоморфны соответствующим графам, приведенным на рис. 3 (пакет “Rgraphviz” накладывает определенные ограничения на визуализацию графов, поэтому получить визуально идентичные графы невозможно). Формальное доказательство изоморфизма будет проведено в будущих работах.

Для выполнения действий алгоритма на языке R над исходным графом были заданы следующие атрибуты вершин графов:

- $GlobalType = \{instruction, data\}$ — тип вершины (оператор, данные);
- $TypeOfData = \{simple, structure\}$ — тип данных вершины, либо тип данных, обрабатываемых оператором в вершине (примитивный, структурный);
- $TypeOfNode = \{datanode, terminal, assignment, ComputeAnd Condition, mergepoint, splitpoint, send, receive, tag, TEMP\}$ — тип вершины в графе (обычная вершина данных, терминальная вершина, вершина присваивания, вершина вычисления условия и ветвления, вершина слияния потоков управления, вершина ветвления, вершина передачи данных другому процессору, вершина получения данных от другого процессора, вершина тега, вершина временных данных примитивного типа).



а



б

Рис. 4. Графы G_{AI} (*а*) и G_{AS} (*б*), полученные с помощи пакетов “graph” и “Rgraphviz” (светлые стрелки соответствуют потоку управления, прямоугольные вершины – данным; серым цветом окрашены вершины структур данных и вершины обработки структур данных)

Каждому шагу методики соответствует собственная функция на языке R. В связи с ограниченностью функционала обработки графов в языке R (отсутствие некоторых видов предикатов инцидентности и смежности) каждая функция получалась более сложной, чем шаг в исходной методике.

Полученные в результате выполнения алгоритма графы на языке R представлены в аналитической форме как объекты класса *GraphNEL*. На основе этих описаний могут быть получены описания программ в исходном коде для ЦП и СП ЭВМ МКОД. Таким образом, приведенная в настоящей работе методика позволяет на высоком уровне абстракции организовать автоматическое распараллеливание последовательного алгоритма, представленного информационным графом, на две части, которые могут быть выполнены параллельно процессорами ЭВМ МКОД.

Заключение. В рамках статьи была формально поставлена задача разделения последовательной программы на две части, которые могут выполняться на ЭВМ МКОД. Приведенная методика декомпозиции информационного графа последовательной программы позволяет автоматически получить граф арифметико-логической обработки данных и граф обработки структур данных, а также определить места в графах, соответствующие обмену данными между ЦП и СП. Эта методика может быть реализована в рамках объектно-ориентированного подхода, как было показано на примере языка R. Выявлены основные недостатки описанной методики декомпозиции.

Несмотря на то, что методика позволяет получить части программы для ЦП и СП ЭВМ МКОД, за рамками рассмотрения остались многие детали: особенности распределения памяти, распараллеливание циклов, поиск оптимального числа передач данных ЦП- и СП-частям программы (и соответствующим графам) и т.д. Указанные проблемы планируется решать в будущих работах. Тем не менее разработанная методика является основой организации параллельной обработки данных для ЭВМ МКОД. Дальнейшее совершенствование методики и решение проблем связи с аппаратной частью позволит применять ЭВМ МКОД для решения полноценных прикладных задач.

ЛИТЕРАТУРА

1. *Clements A*. Principles of Computer Hardware. OUP Oxford, 2006. 672 p.
2. *Попов А.Ю*. Электронная вычислительная машина с многими потоками команд и одним потоком данных. Пат. № 71016. Российская Федерация. 2008. Бюл. № 5.
3. *Попов А.Ю.* Электронная вычислительная машина с аппаратной поддержкой операций над структурами данных // Аэрокосмические технологии. 2009. Т. 1. Тр. Второй Междунар. научно-техн. конф., посвященной 95-летию со дня рождения академика В.Н. Челомея. ВПК “НПО машиностроения”. МГТУ им. Н.Э. Баумана. Москва, 2012. С. 296–301.
4. *Попов А.Ю*. Применение вычислительных систем с многими потоками команд и одним потоком данных для решения задач оптимизации // Инженерный журнал: наука и инновации. 2012. Вып. 1. URL: http://engjournal.ru/catalog/it/hidden/80.html
5. *Попов А.Ю.* О реализации алгоритма Форда – Фалкерсона в вычислительной системе с многими потоками команд и одним потоком данных // Наука и образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2014. № 9. С. 162–180. URL: http://technomag.bmstu.ru/doc/726416.html DOI: 10.7463/0914.0726416

6. *Подольский В.Э.* Об организации параллельной работы некоторых алгоритмов поиска кратчайшего пути на графе в вычислительной системе с многими потоками команд и одним потоком данных // Наука и образование. МГТУ им. Н.Э. Баумана. 2015. № 4. URL: http://technomag.bmstu.ru/doc/764268.html
7. *Овчинников В.А.* Графы в задачах анализа и синтеза структур сложных систем М.: Изд-во МГТУ им. Н.Э. Баумана, 2014. 423 с.
8. *Овчинников В.А.*, *Иванова Г.С.* Методика формального синтеза комбинированных структур данных для представления графов // Инженерный журнал: наука и инновации. 2012. № 1. С. 135–145. URL: http://engjournal.ru/articles/79/79.pdf DOI: 10.18698/2308-6033-2012-1-79
9. *Овчинников В.А.*, *Иванова Г.С.* Оптимизирующие преобразования алгоритмов, использующие свойства множеств, предикатов и операций над нами // Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение. 2013. № 4. С. 53–66.
10. *Gentleman R.*, *Whalen E.*, *Huber W.*, *Falcon S.* Graph: graph: A package to handle graph data structures. R package version 1.44.1. Technical document. May 2015. URL: http://www.bioconductor.org/packages/release/bioc/manuals/graph/man/graph.pdf (дата обращения: 25.05.2015).
11. *Hansen K.D.*, *Gentry J.*, *Long L.*, *Gentleman R.*, *Falcon S.*, *Hahne F.*, *Sarkar D.* Rgraphviz: Provides plotting capabilities for R graph objects. R package version 2.10.0. Technical document. May 2015. URL: http://master.bioconductor.org/packages/release/bioc/manuals/Rgraphviz/man/Rgraphviz.pdf (дата обращения: 25.05.2015).

REFERENCES

[1] Clements A. Principles of Computer Hardware. OUP Oxford, 2006. 672 p.
[2] Popov A.Yu. Elektronnaya vychislitel'naya mashina s mnogimi potokami komand i odnim potokom dannykh [Computer with Multiple Instruction Streams and Single Data Stream]. Pat. no. 71016. Russian Federation, 2008. Bull. no. 5.
[3] Popov A.Yu. Computer with Hardware Support for Operations on Data Structures. *Aerokosmicheskie tekhnologii*, 2009. T. 1. *Tr. Vtoroy Mezhdunar. nauch.-tekhn. konf., posvyashchennoy 95-letiyu so dnya rozhdeniya akademika V.N. Chelomeya*. OAO "VPK "NPO mashinostroeniya" [Aerospace Technologies. Proceedings of the 2nd International scientific and technical conference dedicated to the 95th anniversary of the birth of academician V.N. Chelomey]. MIC "NPO Mashinostroyeniya". Vol. 1. 2009. Bauman MSTU. Moscow, 2012, pp. 296–301 (in Russ.).
[4] Popov A.Yu. Application of Computing Systems with Multiple Instruction Streams and Single Data Stream for Solving Optimization Problems *Jelektr. nauchno-tekh. izd. "Inzhenernyy zhurnal: nauka i innovacii"* [El. Sc.-Tech. Publ. "Eng. J.: Science and Innovation", 2012, iss. 1. Available at: http://engjournal.ru/catalog/it/hidden/80.html
[5] Popov A.Yu. On the Implementation of the Ford-Fulkerson Algorithm in the Multiple Instructions and Single Data Computer System. *Nauka i obrazovanie*. *MGTU im. N.E. Baumana* [Science & Education of the Bauman MSTU. Electronic Journal], 2014, no. 9. Available at: http://technomag.bmstu.ru/doc/726416.html DOI: 10.7463/0914.0726416
[6] Podol'skii V.E. On the Organization of Parallel Operation of Some Algorithms for Finding the Shortest Path on a Graph on a Computer System with Multiple Instruction Stream and Single Data Stream. *Nauka i obrazovanie*. *MGTU im. N.E. Baumana* [Science & Education of the Bauman MSTU. Electronic Journal], 2015, no. 4. Available at: http://technomag.bmstu.ru/doc/764268.html DOI: 10.7463/0415.0764268

[7] Ovchinnikov V.A. Grafy v zadachakh analiza i sinteza struktur slozhnykh system [Graphs in problems of analysis and synthesis of complex system structures]. Moscow, MGTU im. N.E. Baumana Publ., 2014. 423 p.

[8] Ovchinnikov V.A., Ivanova G.S. Method for Formal Synthesis of Combined Data Structures for Graph Representation. *Jelektr. nauchno-tekh. izd. "Inzhenernyy zhurnal: nauka i innovacii"* [El. Sc.-Tech. Publ. "Eng. J.: Science and Innovation", 2012, iss. 1. Available at: http://engjournal.ru/articles/79/79.pdf DOI: 10.18698/2308-6033-2012-1-79

[9] Ovchinnikov V.A., Ivanova G.S. Optimizing Transformations of Algorithms using Properties of Sets, Predicates, and Operations over Them. *Vestn. Mosk. Gos. Tekh. Univ. im. N.E. Baumana, Priborostr.* [Herald of the Bauman Moscow State Tech. Univ., Instrum. Eng.], 2013, no. 4 (93), pp. 53–66 (in Russ.).

[10] Gentleman R., Whalen E., Huber W., Falcon S. Graph: graph: A package to handle graph data structures. R package version 1.44.1. *Technical document*. May 2015. Available at: http://www.bioconductor.org/packages/release/bioc/manuals/graph/man/graph.pdf (accessed 25.05.2015).

[11] Hansen K.D., Gentry J., Long L., Gentleman R., Falcon S., Hahne F., Sarkar D. Rgraphviz: Provides plotting capabilities for R graph objects. R package version 2.10.0. *Technical document*. May 2015. Available at: http://master.bioconductor.org/packages/release/bioc/manuals/Rgraphviz/man/Rgraphviz.pdf (accessed 25.05.2015).

Статья поступила в редакцию 17.07.2015

Подольский Владимир Эдуардович — аспирант кафедры "Компьютерные системы и сети" МГТУ им. Н.Э. Баумана, аналитик ИБС Софт, департамент по работе со сферой образования.
МГТУ им. Н.Э. Баумана, Российская Федерация, 105005, Москва, 2-я Бауманская ул., д. 5.
ИБС Софт, Российская Федерация, 127018, Москва, ул. Складочная, д. 3, стр. 1.

Podolsky V.E. — post-graduate student of Computer Systems and Networks department, Bauman Moscow State Technical University, Analyst at IBS Soft, Education Sector Department.
Bauman Moscow State Technical University, 2-ya Baumanskaya ul. 5, Moscow, 105005 Russian Federation.
IBS Soft, Skladochnaya ul. 3/1, Moscow, 127018 Russian Federation.

Попов Алексей Юрьевич — канд. техн. наук, доцент кафедры "Компьютерные системы и сети".
МГТУ им. Н.Э. Баумана, Российская Федерация, 105005, Москва, 2-я Бауманская ул., д. 5.

Popov A.Yu. — Cand. Sci. (Eng.), Assoc. Professor of Computer Systems and Network department, Bauman Moscow State Technical University.
Bauman Moscow State Technical University, 2-ya Baumanskaya ul. 5, Moscow, 105005 Russian Federation.

Просьба ссылаться на эту статью следующим образом:

Подольский В.Э., Попов А.Ю. Методика декомпозиции информационного графа программы для организации параллельной обработки данных на ЭВМ МКОД // Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение. 2016. № 1. С. 112–128.
DOI: 10.18698/0236-3933-2016-1-112-128

Please cite this article in English as:

Podolsky V.E., Popov A.Yu. Graph decomposition for parallel data processing on MISD computer. *Vestn. Mosk. Gos. Tekh. Univ. im. N.E. Baumana, Priborostr.* [Herald of the Bauman Moscow State Tech. Univ., Instrum. Eng.], 2016, no. 1, pp. 112–128.
DOI: 10.18698/0236-3933-2016-1-112-128