Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Робототехники и комплексной автоматизации»

КАФЕДРА «Системы автоматизированного проектирования (РК-6)»

РАСЧЁТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

по дисциплине «Методы оптимизации»

на тему

«Разработка процедуры обхода при реализации графоориентированного подхода»

Студент РК6-71Б (группа) _____ (подпись, дата) Тришин И.В. (ФИО)

Руководитель КП _____ (подпись, дата) Соколов А.П. (ФИО)

Москва, 2021

Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой *РК-6*
индекс

_____ *А.П. Карпенко*

«____» _____ 2021 г.

ЗАДАНИЕ

на выполнение курсового проекта

Студент группы: РК6-71Б

Тришин Илья Вадимович

(фамилия, имя, отчество)

Тема курсового проекта : Разработка процедуры обхода при реализации графоориентированного подхода

Источник тематики (кафедра, предприятие, НИР): кафедра

Тема курсового проекта утверждена на заседании кафедры «Системы автоматизированного проектирования (РК-6)», Протокол № _____ от «____» _____ 2021 г.

Техническое задание

Часть 1. *Аналитический обзор литературы.*

В рамках аналитического обзора литературы должны быть освещены современные подходы к организации вычислительных процессов при программной реализации сложных вычислительных методов. Отдельное внимание следует уделить графоориентированному подходу.

Часть 2. *Разработка архитектуры программной реализации*

Должен быть разработан и подробно описан алгоритм, позволяющий выполнять операции обработки данных в соответствии с логикой вычислительного метода, описанной в виде ориентированного графа преобразования данных. При потребности в дополнительных структурах данных, используемых в данном алгоритма следует описать и их.

Оформление курсового проекта :

Расчетно-пояснительная записка на 17 листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.):

количество: 8 рис., 0 табл., 8 источн.
[здесь следует ввести количество чертежей, плакатов]

Дата выдачи задания «*07*» октября 2021 г.

Студент _____ Тришин И.В.
подпись, дата ФИО

Руководитель курсового проекта _____ Соколов А.П.
подпись, дата ФИО

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

РЕФЕРАТ

курсовой проект : 17 с., 9 глав, 8 рис., 0 табл., 8 источн.

ОРИЕНТИРОВАННЫЙ ГРАФ, ЧИСЛЕННЫЕ МЕТОДЫ, НАУЧНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ.

Данная работа связана с процессом разработки новой версии программной реализации т.н. «графоориентированного подхода». В ней рассматриваются различные системы, позволяющие организовать отдельные вычислительные процессы в рамках реализации вычислительных методов. Кроме того, описывается разработанный алгоритм выполнения вычислительных процессов в соответствии с логикой вычислительного метода, описанной при помощи ориентированного графа по методологии «графоориентированного подхода». Описываются разработанные структуры данных, необходимые для сопровождения разработанного алгоритма.

Тип работы: курсовой проект .

Тема работы: *«Разработка процедуры обхода при реализации графоориентированного подхода»*.

Объект исследования: подходы к организации вычислительных процессов при реализации сложных вычислительных методов.

Основная задача, на решение которой направлена работа: разработка программной архитектуры параллельного обхода графовых моделей в программном комплексе GBSE.

Цель работы – разработать стратегию обхода графовых моделей в программном каркасе comsdk и предложить программную архитектуру, ей соответствующую

В результате выполнения работы: 1) рассмотрены различные подходы к организации вычислительных процессов при разработке программного обеспечения, реализующего методы решения научно-технических задач; 2) изучен т.н. «графоориентированный подход»; 3) сформированы требования к программной реализации процедуры обхода ориентированного графа, получаемого в процессе описания вычислительного метода с применением “графоориентированного подхода”. 4) разработана архитектура программного модуля, реализующего данную процедуру.

СОКРАЩЕНИЯ

GBSE Graph-base Software Engineering. 8

ПО Программное обеспечение. 7

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современные научно-технические исследования зачастую включают в себя задачи, решения которых требуют большого количества вычислений. Для упрощения процесса решения используется или разрабатывается различное программное обеспечение (ПО). Например, могут применяться программные средства, предоставляющие универсальный базовый математический аппарат (MathCAD, Maple, Mathematica и т.п.). Как правило, подобные программные продукты предоставляют некоторый формальный язык описания математических выражений или редактор формул. Преимущества и недостатки множества подобных программ освещены в [1].

При всех преимуществах применения подобных программ при решении сложных вычислительных задач за пользователем остаётся необходимость формулировать их математические постановки (т.е. формировать математические модели, составлять системы уравнений и т.д.). Зачастую требуется решать множество задач с схожей постановкой, но с различными входными параметрами (как, например, при анализе прочностных характеристик технических объектов). Становится целесообразным разработать автоматизированные средства решения подобных типовых задач. При этом от разработчика требуются глубокие познания в предметной области задачи. Возникает потребность в некоторой промежуточной системе, позволяющей формально поэтапно описать метод решения некоторой задачи для удобства его последующей реализации. В данной работе внимание сосредоточено на подобных системах.

С точки зрения разработки ПО, при описании некоторого вычислительного метода целесообразно выделить в нём отдельные этапы, отдельные операции обработки данных. Каждой такой операции требуются входные данные. По завершении выполнения операции получаются выходные данные. При этом выходные данные одной операции могут являться входными для одной или нескольких других операций. Между ними формируются зависимости по входным и выходным данным. Для учёта этих зависимостей возникает необходимость правильным образом организовать выполнение операций в пределах отдельно взятого метода.

В наши дни популярность приобретает применение научных систем организации рабочего процесса (англ. scientific workflow systems). Такие системы позвояют автоматизировать процессы решения научно-технических задач, предоставляя средства организации и управления вычислительными процессами [2]. Процесс работы с подобными системами состоит из 4 основных этапов:

1. составление описания операций обработки данных и зависимостей между ними;
2. распределение процессов обработки данных по вычислительным ресурсам;
3. выполнение обработки данных;
4. сбор и анализ результатов и статистики.

Примерами подобных систем могут служить Pegasus[3], Kepler[4] и pSeven[5].

Одной из ключевых особенностей подобного подхода к реализации решений научно-технических задач является выделение операций обработки данных в отдельные программные модули (функции, подпрограммы, скрипты). При известных входных и выходных данных каждого модуля становится возможной их независимая разработка[6]. Это позволяет распределить их разработку между несколькими людьми. Вследствие этого уменьшается объём работы по написанию исходных кодов, приходящийся на одного исследователя. Это в свою очередь облегчает отладку и написание документации, что положительно сказывается на общем качестве реализуемого ПО.

В основном, в научных системах организации рабочего процесса для описания связей между отдельными операциями обработки данных используются ориентированные графы. Помимо описания связей между вычислительными процессами ориентированные графы также находят применение при планировании деятельности (сетевые графики, граф-схемы). В научно-технической среде большее распространение получили сети Петри, диаграммы потоков данных (DFD) и диаграммы перехода состояний. В данной работе основное внимание отводится т.н. “графоориентированному подходу”(GBSE)[7] и его реализации.

1 Постановка задачи

1.1 Концептуальная постановка задачи

Определение 1. *Графовой моделью* вычислительного метода назовём совокупность операций обработки данных, которые включает в себя данный метод, и ориентированный граф, определяющий очерёдность и логику выполнения обозначенных операций.

Определение 2. *Ветвью* графовой модели назовём подграф, состоящий из последовательно соединённых друг с другом вершин, такой, что из каждой вершины выходит не более одного ребра.

Пример ветви графа выделен на рисунке 1 красным цветом.

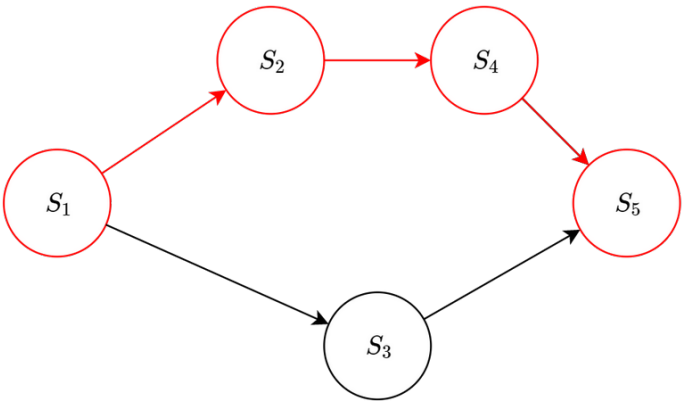

Рисунок 1. Пример графа с выделенной ветвью

Помимо этого целесообразно обозначить некоторые ключевые особенности логики «графоориентированного подхода».

1) Узлам графовой модели ставятся в соответствие т.н. «состояния данных» (см. далее).
2) *Состоянием данных* называется набор поименованных переменных фиксированного типа (скалярного или векторного), характерных для конкретного этапа описываемого метода.
3) Рёбрам графовой модели ставятся в соответствие т.н. функции перехода, осуществляющие преобразования данных из одного состояния в другое.
4) Функции перехода имеют два ключевых компонента: функцию-предикат и функцию-обработчик (см. далее).
5) *Функцией-обработчиком* f называется функция, непосредственно преобразующая данные из одного состояния в другое.
6) *Функция-предикат* p ставит в соответствие входному набору данных 1 или 0 согласно внутренней логике. Функция-обработчик выполняется только в случае, когда функция-предикат возвращает значение 1. Ситуация, когда функция-предикат возвращает 0 считается исключительной и обрабатывается отдельно.

«Графоориентированный подход» подразумевает параллельное независимых ветвей графа. На рисунке 2 после выполнения рёбер F_{12} и F_{13} будет получено два независимых состояния данных S_2 и S_3 соответственно. Далее возникает задача правильным образом перевести данные из этих состояний в общее состояние S_4.

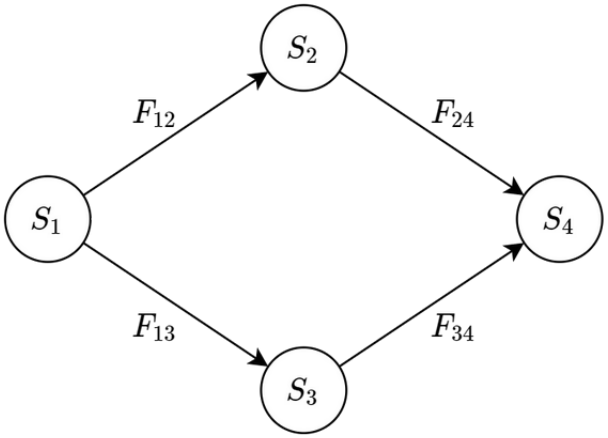

Рисунок 2. Пример графовой модели, требующей параллельного исполнения

Данный подход значительно увеличивает эффективность использования ресурсов вычислительной системы и ускоряет процесс решения, однако добавляет некоторые второстепенные задачи при разработке. Так в примере на рисунке 2 рёбра F_{12} и F_{13} выполнялись параллельно, а значит полученные в результате их выполнения данные существуют в самом общем случае в различных адресных пространствах оперативной памяти (возможно даже на двух разных вычислительных машинах). В момент разветвления графа должно происходить корректное предоставление вычислительным ресурсам (потокам, процессам, узлам кластера и т.п.) доступа к обрабатываемым данным. Помимо этого алгоритм обхода графовой модели должен корректно отрабатывать слияние ветвей графа и в частности при необходимости выполнять сбор данных.

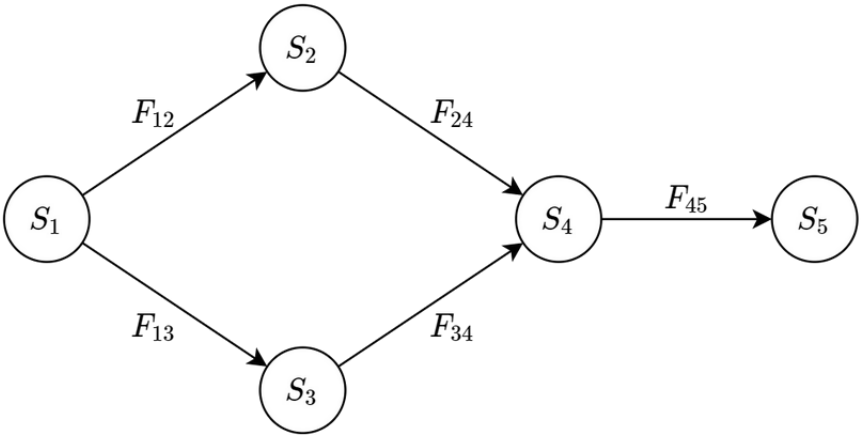

Рисунок 3. Пример графовой модели с совмещением ветвей

На рисунке 3 ветви $S_1 \to S_2 \to S_4$ и $S_1 \to S_3 \to S_4$ выполняются с использованием различных вычислительных ресурсов, но ребро F_{45} выполняется в пределах одной «общей» ветви графа $S_4 \to S_5$, и в момент его выполнения ресуры, выделенные на выполнение двух

параллельных ветвей уже не требуются. Таким образом, целесообразно разработать управляющую структуру, которая бы отвечала за выделение и освобождение вычислительных ресурсов во время работы с несколькими параллельными ветвями графа.

Кроме того, разрабатываемая архитектура должна поддерживать несколько вариантов параллельного исполнения. Среди прочих желательна поддержка:

– поочерёдного выполнения (в первую очередь для отладки) в одном потоке управления;
– выполнения с использованием нескольких процессов операционной системы;
– выполнения с использованием нескольких потоков процессора;
– выполнения на удалённых узлах (через SSH-соединение).

Таким образом целесообразна поддержка единого интерфейса обозначенной управляющей структуры для разных режимов выполнения (последовательный, параллельный, распределённый и пр.).

В случае, когда параллельного выполнения не требуется и предусмотрено условное ветвление, оно должно быть реализовано при помощи специальных функций, привязываемых к узлам графовой модели. В контексте «графоориентированного подхода» такие функции называются *селекторами*. Формально функция-селектор h_i, привязанная к узлу v_i, должна входному набору данных $\bar{D}$ ставить в соответствие множество рёбер E_i, выходящих из v_i, проход по которым нужно совершить. Пример работы функции-селектора демонстрирует рисунок 4.

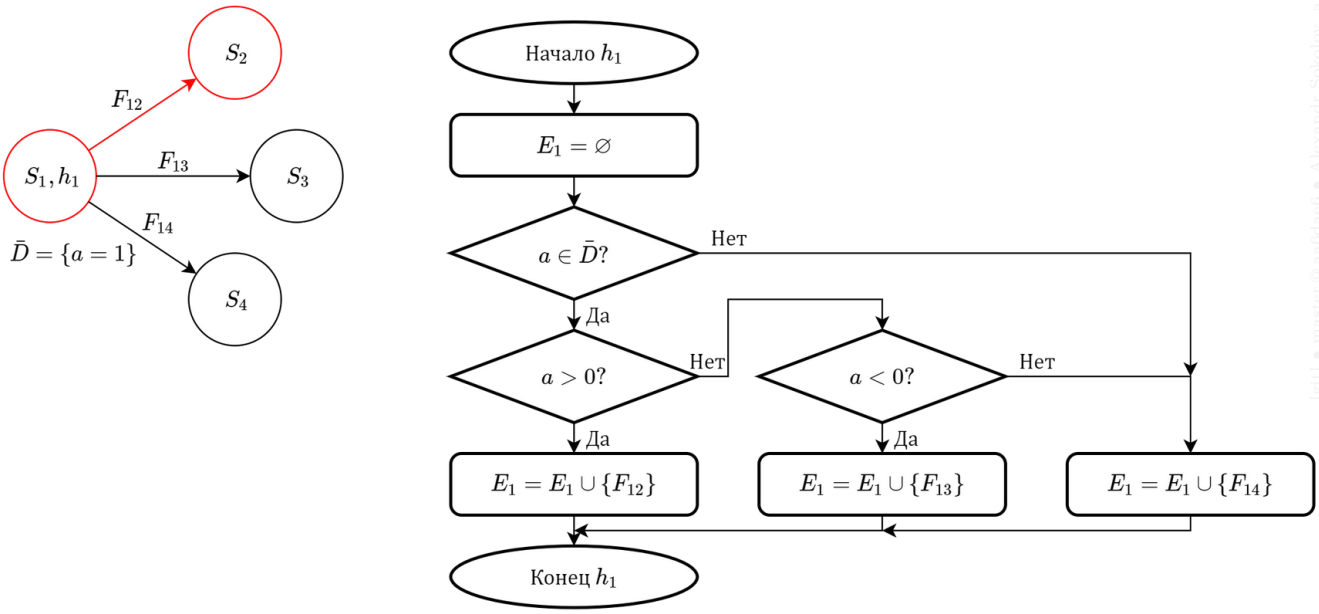

Рисунок 4. Пример фрагмента графовой модели с функцией-селектором

На данном рисунке красным обозначено ребро, переход по которому будет совершён после вызова функции-селектора.

Алгоритмы обхода графовых моделей, предусматривающие условное ветвление и параллельное выполнение ветвей представлены в разделе 2.

2 Разработанные алгоритмы

Сначала был разработан упрощённый алгоритм обхода, поддерживающий только условное ветвление. Его блок-схема представлена на рисунке 5.

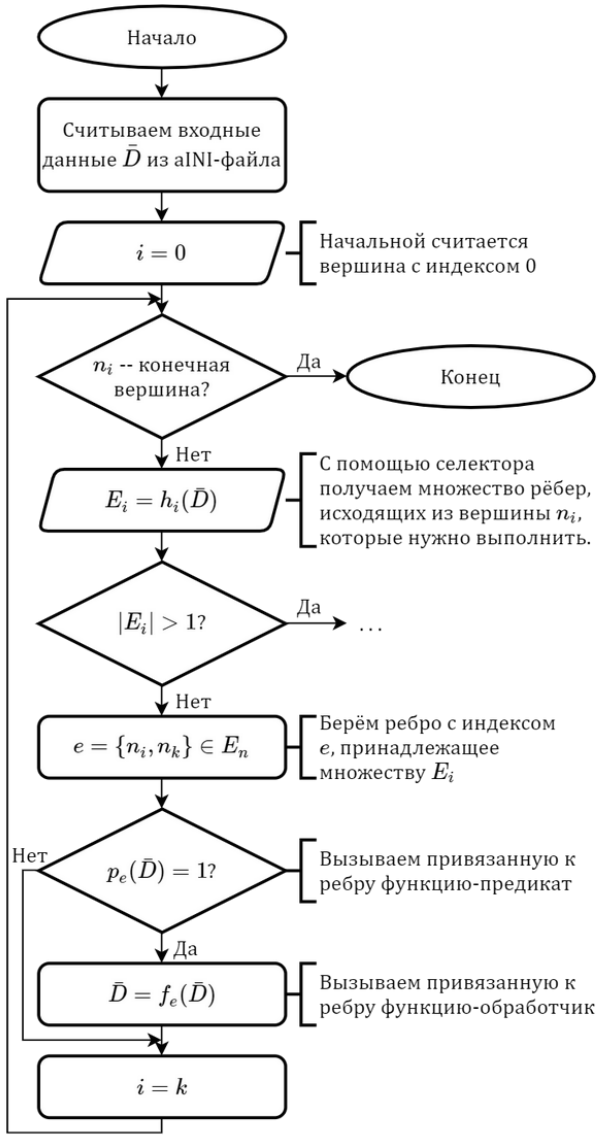

Рисунок 5. Блок-схема алгоритма обхода графовой модели, не предполагающей параллельное исполнение

Далее был отдельно рассмотрен случай, когда необходимо параллельно обойти несколько ветвей. Для контроля за параллельным исполнением функций перехода и выполнения необходимых операций по распределению и сбору данных с задействованных вычислительных ресурсов была разработана управляющая струкутура «контейнер выполнения». Более подробно структура описана в разделе 3. Логика работы с данной структурой представлена на рисунке 6.

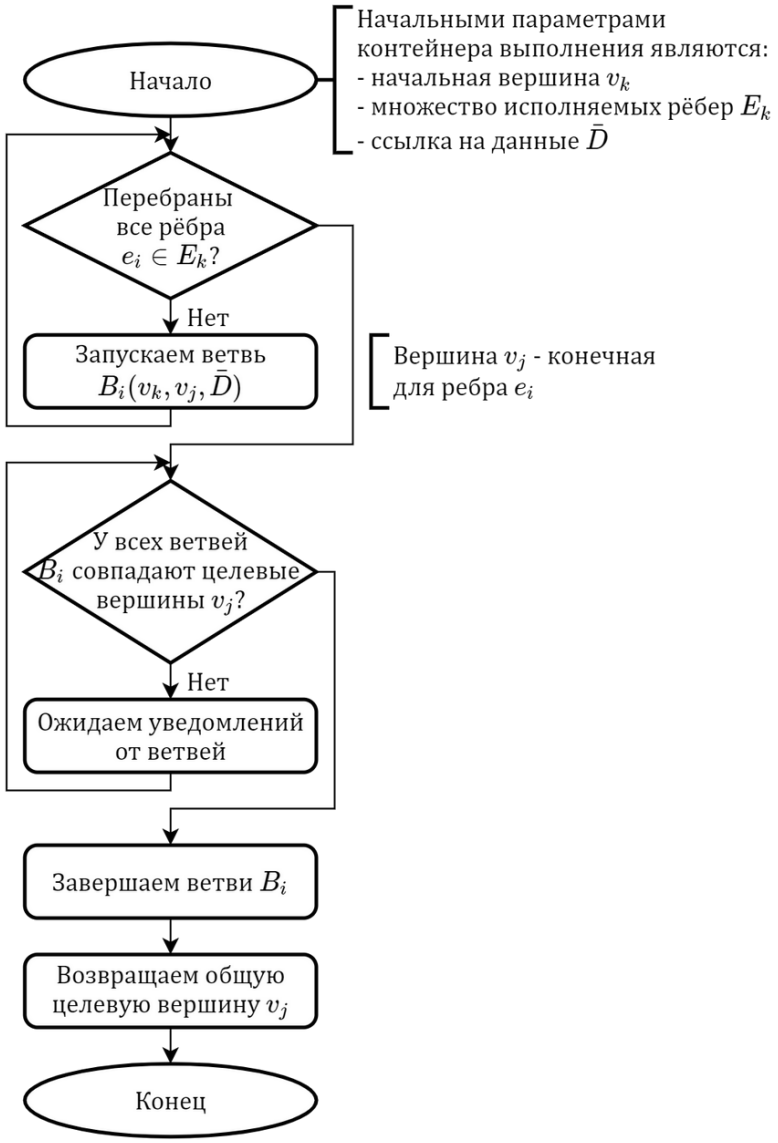

Рисунок 6. Блок-схема алгоритма отслеживания параллельного исполнения ветвей графа

При этом внутри управляющей структуры «контейнер выполнения» подразумевается создание отдельных «ветвей».

Алгоритм обхода одной ветви представлен на рисунке 7.

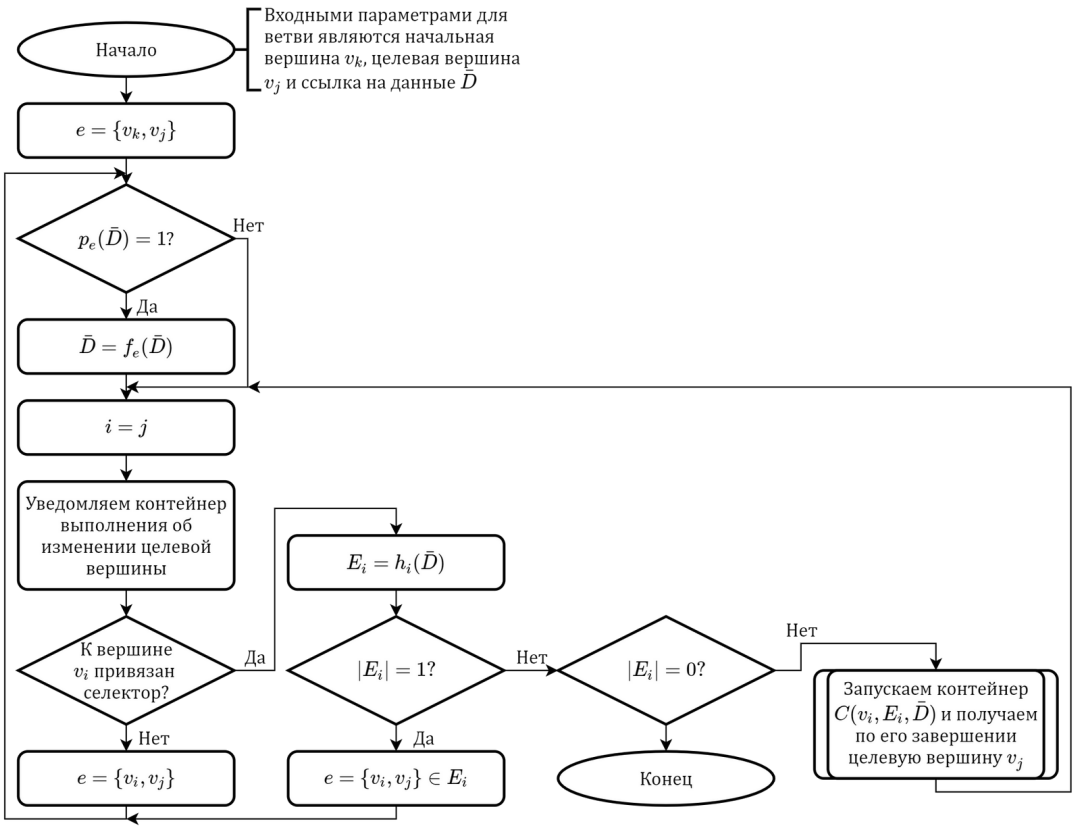

Рисунок 7. Блок-схема алгоритма исполнения одной из параллельных ветвей

Общий алгоритм предполагает завершение выполнения ветви по сигналу от «контейнера выполнения». Ситуация, когда по какой-то причине для текущей вершины v_i и данных $\bar{D}$ не было выбрано ни одного ребра, является исключительной и должна обрабатываться отдельно.

Разработанные алгоритмы будут реализованы в ходе выполнения практической части ВКР с использованием разработанных в рамках курсовых проектов структур данных.

3 Разработанные структуры данных

Основной структурой данных, поддерживающей разработанный в разделе 2 алгоритмы является структура данных «контейнер выполнения». Задача её внутренней логики – контролировать параллельный обход нескольких ветвей графа и отслеживать их слияние. Помимо прочего это подразумевает выделение и освобождение вычислительных ресурсов для параллельного обхода. Поскольку в общем случае могут применяться различные вычислительные ресурсы (потоки ядер процессора, процессы операционной системы, узлы вычислительного кластера и пр.), было принято решение разработать единый интерфейс структуры «контейнер выполнения» без привязки к конкретным ресурсам. Это позволит независимо разрабатывать различные реализации данной структуры для различных режимов параллельного обхода.

Помимо прочего при проектировании интерфейса структуры данных «контейнер выполнения» были задействованы разработанные в рамках курсового проекта по дисциплине «Модели и методы анализа проектных решений» структуры данных «операция с вершиной» (NodeOp) и «операция с ребром» (EdgeOp)[8].

Интерфейсы спроектированных структур данных были описаны с помощью UML-диаграмм, представленных на рисунке 8. Были учтены особенности описания классов на языке C++.

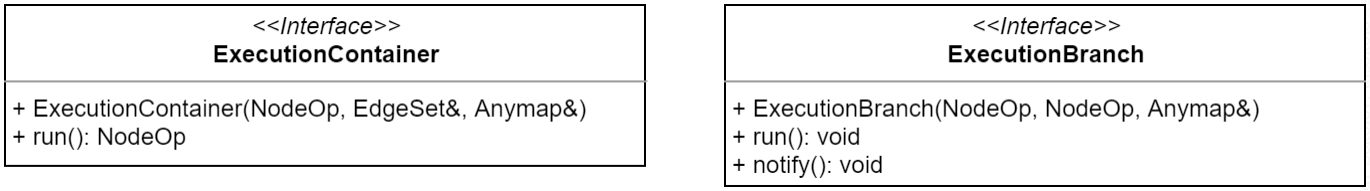

Рисунок 8. UML-диаграммы разработанных структур данных

Конструктор класса ExecutionContainer соответствует алгоритму, описанному в разделе 2. Метод run выполняет алгоритм, описанный блок-схемой на рисунке 6.

Кроме того, была спроектирован интерфейс структуры «ветви исполнения» (ExecutionBranch на рисунке 8). Её задача – выполнение обхода одной конкретно взятой ветви (метод run()) графовой модели с уведомлением «контейнера исполнения» о завершении каждой функции перехода (метод notify()), как того требует алгоритм.

Таким образом, разработанные классы реализуют в себе логику параллельного обхода ветвей графовой модели и хранят данные в соответствии с разработанной процедурой обхода.

ЗАКЛЮЧЕНИЕ

Таким образом, в данной работе были изучены различные автоматизированные системы, упрощающие проведение научно-технических исследований и решение возникающих в их ходе задач. Были рассмотрены подходы к организации процессов обработки данных в рассмотренных системах. Среди прочего, был изучен т.н. «графоориентированный подход» и принципы обхода графовых моделей в нём. Для данного подхода был разработан алгоритм обхода, дающий возможность параллельного выполнения независимых процессов обработки. Помимо этого были разработаны структуры данных для сопровождения данного алгоритма в рамках реализации обозначенного подхода в библиотеке comsdk на языке C++.

Список использованных источников

1 List of computer algebra systems [Электронный ресурс]. 2022. Дата обращения: 27.03.2022. URL: https://en.wikipedia.org/wiki/List_of_computer_algebra_systems.

2 Workflows and e-Science: An overview of workflow system features and capabilities / D. E., G. D., S. M. et al. // Future Generation Computer Systems. 2009. Vol. 25, no. 5. P. 528 – 540.

3 Pegasus in the cloud: Science automation through workflow technologies / Deelman E., Vahi K., Rynge M. [и др.] // IEEE Internet Computing. 2016. Т. 20, № 1. С. 70 – 76.

4 Kepler: An extensible system for design and execution of scientific workflows / Altintas I., Berkley C., Jaeger E. [и др.]. Т. 16. 2004. С. 423 – 424.

5 Alexey M. Nazarenko Alexander A. Prokhorov. Hierarchical Dataflow Model with Automated File Management for Engineering and Scientific Applications // Procedia Computer Science. 2015. Т. 66. URL: https://www.sciencedirect.com/science/article/pii/S1877050915034055?pes=vor.

6 Данилов А.М., Лапшин Э.В., Беликов Г.Г., Лебедев В.Б. Методологические принципы организации многопотоковой обработки данных с распараллеливанием вычислительных процессов // Известия вузов. Поволжский регион. Технические науки. 2001. № 4. С. 26–34.

7 Соколов А.П. Першин А.Ю. Графоориентированный программный каркас для реализации сложных вычислительных методов // Программирование. 2018. № X.

8 Тришин И.В. Развитие графоориентированного программного каркаса для реализации сложных вычислительных методов [Курсовой проект]. 2021.

Выходные данные

Тришин И.В.. Разработка процедуры обхода при реализации графоориентированного подхода по дисциплине «Методы оптимизации». [Электронный ресурс] — Москва: 2021. — 17 с. URL: https://sa2systems.ru:88 (система контроля версий кафедры РК6)

Постановка: *канд. физ.-мат. наук, Соколов А.П.*

Решение и вёрстка: *студент группы РК6-71Б, Тришин И.В.*

2021, осенний семестр