Содержание

Введение

1. Постановка задачи

2. Анализ исходных данных

3. Описание используемой структуры ВС

4. Описание алгоритма решения задачи

4.1 Основные определения

4.2 Алгоритм построения нитей в сети G

4.3 Алгоритм уплотнения нитей

4.4 Алгоритм распределения вершин графа решаемой задачи на узлах вычислительной сети с одинаковой степенью вершин

5. Описание интерфейса программы

6. Результаты работы программы

Заключение

# Введение

В настоящее время увеличилась тенденция использования многопроцессорных систем для обработки данных. Для эффективного использования таких систем необходимо, во-первых, преобразовывать последовательные алгоритмы обработки данных в параллельные, а во-вторых, использовать специальные алгоритмы (планировщики), которые позволят распределить операторы параллельных алгоритмов по процессорам вычислительной сети. По своей сути, планировщик является частью основного алгоритма и служит для обеспечения эффективного выполнения основного алгоритма в условиях конкретной ВС.

При этом эти алгоритмы-планировщики могут использовать различные критерии оптимизации. Например, для получения такого распределения, при котором максимально эффективно будут использоваться все процессоры ВС или для получения такого распределения, при котором заданный алгоритм будет решаться за минимальное время при минимизации числа процессоров.

Разработка подобных алгоритмов связана с рядом трудностей. В частности, необходимо проанализировать большое количество условий, учесть множество различных ситуаций, которые возникают при распределении операторов по нитям и нитей по процессорам ВС. Кроме того, необходимы точные исходные данные, такие как времена расчета отдельно взятых операторов, объем передаваемых данных между ними. Необходимо также знать времена передачи данных между процессорами в структуре ВС.

Решение задачи создания таких алгоритмов и их последующее применение увеличит быстродействие обработки данных на многопроцессорных системах.

# 1. Постановка задачи

Цель - найти оптимальный план распределения решаемой задачи по узлам вычислительной сети (ВС).

Для достижения этой цели необходимо разработать алгоритм распределения вершин информационного графа по процессорам заданной структуры вычислительной сети. В результате этого распределения исходная задача должна решаться за минимально возможное время на ВС. Число процессоров при этом должно быть минимизировано с учетом обеспечения решения задачи за минимальное время.

Далее необходимо реализовать полученный алгоритм в виде программы. Программа должна обеспечить:

1. Возможность получения матрицы следования для различных информационных графов. Это позволит лучше протестировать полученный алгоритм распределения.

2. Построение информационного графа решаемой задачи, по его заданной матрице следования.

3. Построение диаграммы размещения нитей на узлах ВС.

4. Построение временной диаграммы выполнения алгоритма.

# 2. Анализ исходных данных

Решаемые задачи представляются треугольными матрицами следования 40х40 со скалярными весами для ИГ с помощью датчика псевдослучайных чисел, где первые две строки - нулевые. Необходимо сформировать ИЛГ с тремя логическими операторами в графе, представленном полученной матрицей, заменив произвольные вершины графа логическими. Связи, исходящие из логических операторов, кроме двух, исключить.

Рассматриваемая структура вычислительной сети - обобщенный гипертор. Степень вершины графа решаемой задачи не должна превышать 6, при этом она превышает степень вершины ВС на 3. В случае невыполнения этого условия матрица следования должна корректироваться: из столбца удаляются последние единицы, а из строки - первые.

Необходимо минимизировать количество узлов ВС, обеспечивая при этом минимальное время решения задачи.

С помощью программных средств необходимо организовать просмотр процесса построения нитей. Веса дуг и вершин графа решаемой задачи определяются с помощью датчика псевдослучайных чисел. Вес вершины (pV) - время расчета (в условных единицах) оператора на i-ом процессоре.

Вес дуги (pA) - время передачи (в условных единицах) данных из одного оператора в другой, при условии, что эти операторы находятся на соседних процессорах. Если операторы находятся на процессорах, расстояние между которыми равно d, то время передачи данных из одного оператора в другой будет равно d\*pA.

# 3. Описание используемой структуры ВС

Структура ВС типа **обобщенный nD-тор** описывается графом GS (M,S\*), где М - множество вычислителей M={mi}, i=0…N-1, а S\* - сеть, состоящая из множества ребер.

Структура ВС типа обобщенный трехмерный гипертор описывается следующими соотношениями:

по каждой координате k=1, 2, 3 откладываются точки (вершины) с номерами 0,1,…,Nk-1, где Nk - размерность тора по координате k;

множество вершин графа коммутационной структуры задается декартовым произведением [0,1,…,N1-1] x [0,1,…,N2-1] x [0,1,…,N3-1];

две вершины соединяются ребром, если их декартовы произведения отличаются друг от друга на единицу по любой координате или на N1-1 по координате 1 или на N2-1 по координате 2 или на N3-1 по координате 3 соответственно.



Рис.1 - Схема представления обобщенного трехмерного тора 3x2x2

Данная структура ВС имеет ряд преимуществ перед другими структурами ВС такими, как циркулянта, n-мерный двоичный гиперкуб, обобщенный трехмерный гиперкуб, бинарное дерево. Обобщенный гипертор является симметричной структурой с множеством дополнительных связей, что значительно облегчает процесс распределения вершин информационного графа по процессорам заданной структуры вычислительной сети.

# 4. Описание алгоритма решения задачи

# 4.1 Основные определения

**Вершина** - оператор ИЛГ заданной задачи.

**Вес вершины (pV) -** время расчета вершины на i-ом процессоре.

**Время старта вершины (Vs) -** время старта расчета вершины в существующем разбиении вершин между процессорами.

**Время финиша вершины (Vf) -** время финиша расчета вершины в существующем разбиении вершин между процессорами.

**Начальная и конечная вершины** добавляются к информационному графу. Начальная вершина имеет номер 0 и необходима для того, чтобы граф имел одну точку входа. Конечная вершина имеет номер N+1, где N - размерность матрицы следования исходного информационного графа, и необходима, чтобы граф имел одну точку выхода. Веса этих вершин равны нулю. Веса дуг, выходящих из нулевой вершины равны единице. После этого добавления исходная матрица следования S преобразуется, будет иметь размер N+2 и обозначаться C.

**Высота вершины (h) -** максимальное время от начала выполнения вершины до конца выполнения алгоритма, заданного матрицей следования С.



По определению высота конечной вершины равна нулю.

**Родители вершины** - все предшествующие данной вершине вершины, от которых она зависит по данным.

**Нить** - набор из одной или нескольких вершин, которые последовательно рассчитываются на одном процессоре.

микропроцессорная сеть алгоритм планировщик

**Родительские нити вершины** - набор нитей, каждая из которых содержат одного или нескольких родителей данной вершины

**Время готовности вершины (r) -** максимум из времен финиша всех родителей вершины.

**Время старта нити (sT) -** время старта нити в существующем разбиении нитей между процессорами.

**Время финиша нити (fT) -** время финиша нити в существующем разбиении нитей между процессорами.

**Номер процессора нити (nfp) -** номер процессора, на котором рассчитывается нить в существующем разбиении.

## 

# 4.2 Алгоритм построения нитей в сети G

Алгоритм построения нитей в графе G, представляющим решаемую задачу.

1. В графе G выделим множество начальных вершин В матрице S, построенной для графа G начальным вершинам соответствуют нулевые строки. Вычислим i: =1 - параметр определяющий текущий номер элемента в множестве , V - номер массива перебора операторов.



K: =1 - номер очередной создаваемой нити, -множество связей нитей с другими нитями, f: =0 - номер очередного разрезания графа G, - множество продолжения нитей.



1. Возьмем вершину .



1. Вычислим обобщенный вес вершины как если вес вершины не модифицировался или в противном случае.



1. Если из вершины не выходит связь, то переходим к шагу 9, иначе выполняется следующий шаг.



1. Если их вершины выходит одна связь в j вершину, т.е. в матрице S d -м столбце содержится единица в j-й строке. Обобщенный вес j-й вершины определится как если j-я вершина не модифицировалась и в противном случае. Обобщенный вес вершины определяется как на шаге 3. Переходим к шагу 10, иначе выполняется следующий шаг.



1. Если из вершины выходит несколько связей (развертка -вершины), то среди множества дуг J, исходящих из вершины ищем , (1), где -вес дуги, исходящей из вершины и входящей в вершину j.



1. Если условию (1) удовлетворяют несколько вершин, то выбирается первая вершина рассматриваемого множества, составляющих эти вершины. Для вершины вычисляется вес , если вершина не модифицировалась и , в противном случае. Веса вершин из множества J, исключая вершину , вычисляются как , если j-я вершина не модифицировалась и в противном случае, где -вес дуги, выходящей из вершины и входящей в вершину j. Обобщенный вес вершины определяется, как на шаге 3. Переходим к шагу 11. Если из вершины не выходит несколько связей, то выполняется следующий шаг.



1. Если вершина входит в свертку J, то обобщенный вес вершины , связанной с вершиной , вычисляется как , если обобщенный вес -й вершины не модифицировался и в противном случае. Веса остальных вершин, исключая вершину , вычисляются как , если вес вершины j не модифицировался и в противном случае. Обобщенный вес вершины вычисляется как на шаге 3. Переходим на шаг 12.



1. Вершина включается в Tk-ю нить как конец нити и исключается из рассмотрения. Tk-я нить включается в множество нитей NT.



1. Вычислим . Если , то вычисляем f: =f+1 и переходим к шагу 13, иначе положим i: =i+1 и переходим на шаг 2.



1. Вершина оформляется как элемент Tk-й нити и исключается из рассмотрения. Вершина включается в множество продолжения нити . Дуги исключаются из графа G или его компонентов. Составляется таблица (множество) связей фрагмента Tk нити в виде , где -включает номера операторов множества J, исключая оператор . Осуществляется переход на шаг 10.



1. Вершина оформляется как элемент Тк нити и исключается из рассмотрения. Вершина включается в множество продолжения нити . Дуги исключаются из графа G или его компонентов. Составляется таблица связей для фрагмента нити, где -включает номера операторов составляющих множество J, исключая оператор . Осуществляется переход на шаг 10.



1. Рассмотрим множество К компонентов графа G, образованные в результате удаления связей. Если множество , то переходим на шаг 16, иначе выполняем следующий шаг.



1. С помощью матрицы S, составленной для компонентов графа G определим множество начальных вершин .



1. Образуем множество таким образом, чтобы элементы множества предшествовали элементам множества , полученного на шаге 14. Множество Положим i: =1 и переходим на шаг 2.



1. Для графа G\*, который имеет ту же конфигурацию, что и исходный граф G, но в котором изменены весы вершин с учетом весов дуг, вычислим ранние сроки окончания выполнения операторов.
2. Для каждой нити вычислим время старта нити в виде , где - ранний срок выполнения первого оператора Тк нити. Время окончания нити определяется как , где - ранний срок окончания последнего оператора Тк нити.



Конец описания алгоритма.

Таким образом, каждая нить характеризуется своим номером (к), временем начала нити , временем окончания нити и таблицей связей к-й нити с другими операторами, входящими в нити множества Тк.



## 

# 4.3 Алгоритм уплотнения нитей

1. Времена начала и конца нитей объединяются в множества где n - число нитей, полученных в предыдущем алгоритме.



1. Упорядочим множество в порядке не убывания. Элементы представляются таким образом, чтобы i-ый номер начала нити соответствовал i-му номеру конца нити.



1. Найдем такой что для соседних нитей, размещаемых на интервале [0,T], после вычислений по соотношения (1) выполняется условие, что время окончания предшествующей нити должно быть меньше или равно времени начала последующей нити.



1. Если нити, удовлетворяющие условию (1) найдены, то соответствующие и удаляются и записываются в множество в виде составной к-й нити. Объединяются множества таблиц связей в виде где I - число нитей, вошедших в составную нить.



1. Если =0, то работа алгоритма заканчивается, иначе осуществляется переход к п.3.



Конец описания алгоритма.

## 

# 4.4 Алгоритм распределения вершин графа решаемой задачи на узлах вычислительной сети с одинаковой степенью вершин

Предполагаем, что количество узлов вычислительной сети неограниченно и множество нитей Т получено с помощью алгоритма 4.2.

1. Просматриваем множество нитей Т, выберем к-ю нить с максимальным количеством элементов в множестве (таблице связей к-й нити). Предположим таблица связей имеет элементов.



1. Если степень i-й вершины вычислительной сети есть , то сравниваем и .



1. Если , то нить размещается в узле i, и переходим к шагу, иначе следующий шаг.



1. Если то образуется комплексный узел, в котором один вычислитель основной, остальные являются передающими звеньями. Полагаем f: =1 и переходим к следующему шагу.



1. Определяем где Т множество нитей.



1. Если , то переход на шаг 10.



1. Нить занимает узел вычислительной сети на минимально возможном расстоянии от узла i. Все вершины из окружения нити Ti, принадлежащие множеству удаляются из узлов, соседних с узлом I, т.е.



1. В соседних узлах с узлом размещаются элементы таблицы . Если количество узлов с минимальным расстоянием недостаточно, то организуются комплексные узлы, как пункте 4.



1. Если то вычисляется f: =f+1 и переход на шаг 5. Иначе полагаем и переход на шаг 5.



1. Вычисляем узел х, удовлетворяющий соотношению , где - количество свободных связей у рассматриваемых узлов. f=1,…f `, затем полагаем i: =x, T: =T\Tк. Если , то переходим к шагу 1, иначе конец алгоритма.



# 5. Описание интерфейса программы

Интерфейс разработанной программы состоит из одного окна, содержащих несколько вкладок:

1. Вкладка генерации информационного графа алгоритма и результатов выполнения разбиения алгоритма на нити. (рис.2).

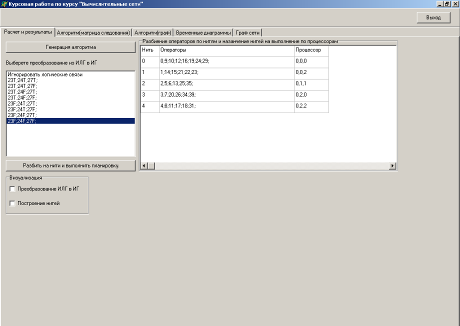


Рис.2. Вкладка управления работой программы

На вкладке можно осуществить генерацию ИЛГ, выбрать метод преобразования ИЛГ в ИГ и задать режим визуализации (какие построения необходимо визуализировать).

2. Вкладка вывода информационного графа, заданного матрицей следования (рис.3).

Содержит сгенерированную матрицу следования алгоритма. В крайнем правом столбце приведены веса операторов, веса связей не приводятся, что бы можно было легче анализировать матрицу визуально.

В матрице следования в столбцах задаются все выходящие из данной вершины связи, а в строках - все входящие в заданную вершину связи. При этом задаваемое значение 1 означает, что связь между операторами есть, если связь пронумерована с буквами ‘T’ или ‘F’, то это соответствующая логическая связь. Если связи между вершинами нет, то позиция не заполняется.

Поскольку ИЛГ по заданию не должен содержать циклов, то данные можно вводить только ниже главной диагонали матрицы следования.

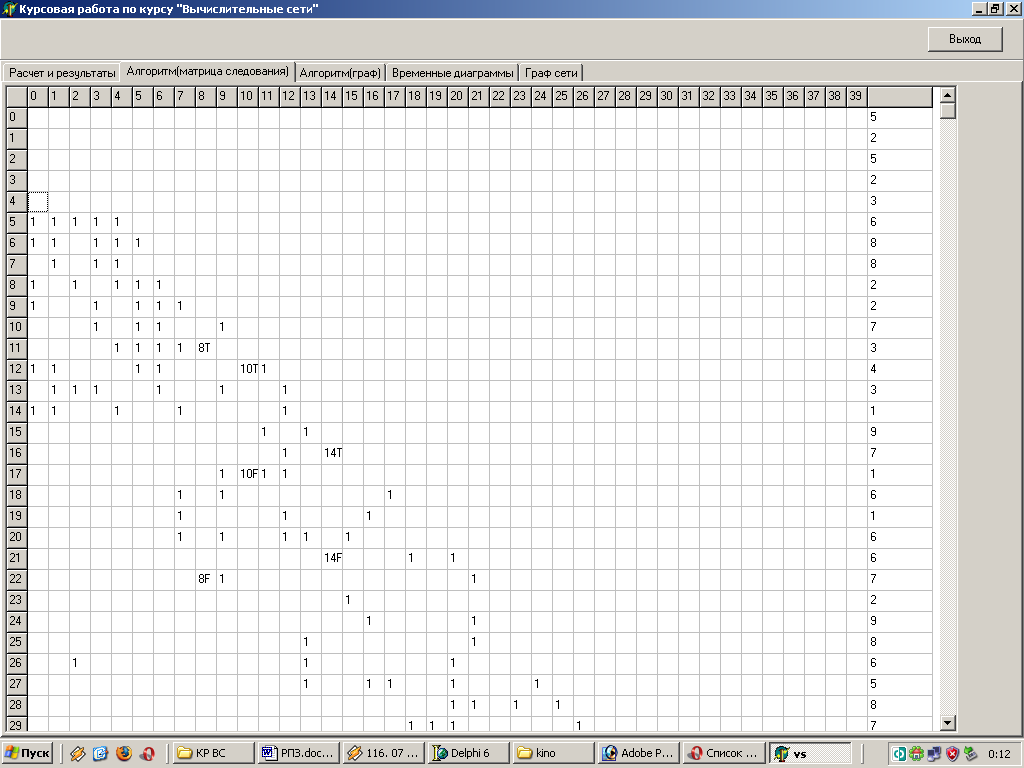


Рис.3 - Окно вывода информационного графа.

3. Окно вывода информационного графа, заданного матрицей следования (рис.4).

Отображает алгоритм в виде графа, что упрощает визуальный анализ. При настройке параметров визуализации показывает процесс построения нитей и преобразования ИЛГ в ИГ.

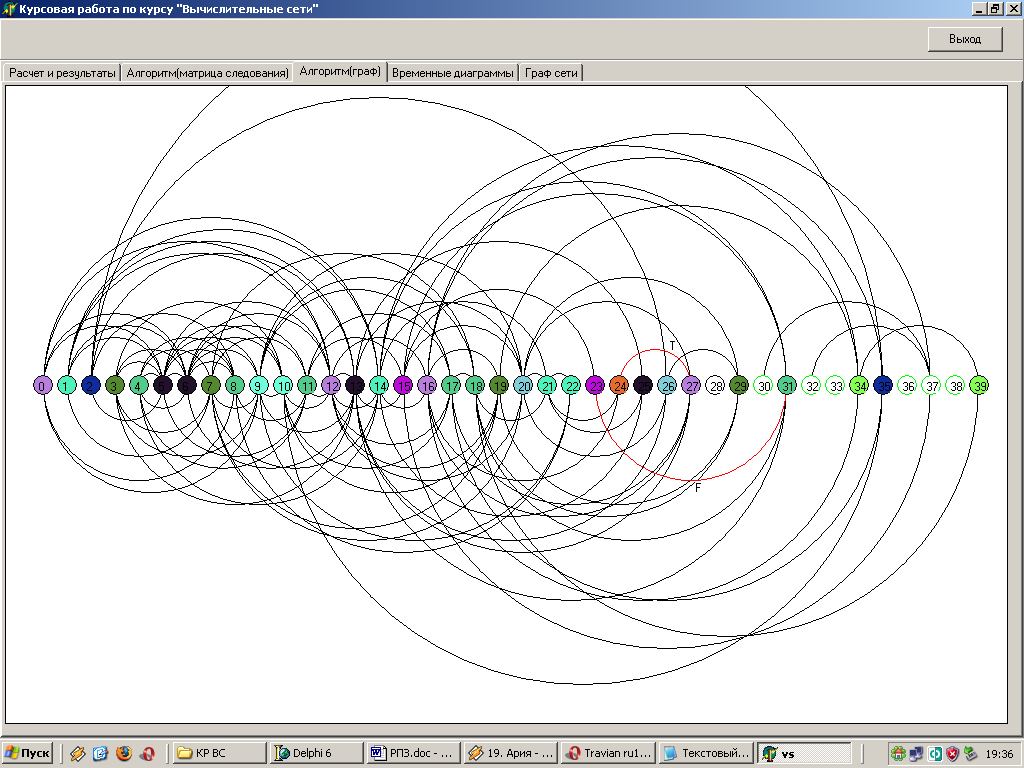


Рис.4 - Вкладка вывода ИЛГ.

4. Временные диаграммы распределения нитей по процессорам (рис.4).

Здесь указывается, какой нити принадлежит оператор, и в какие сроки он выполняется.

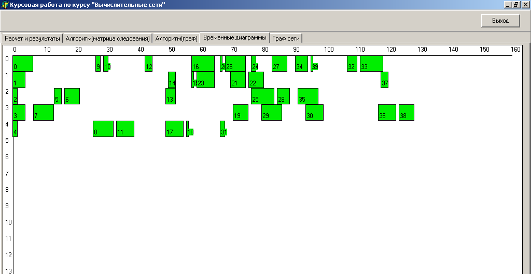


Рис.4. Временные диаграммы нитей.

5. Вкладка распределения нитей по процессорам. Содержит описание структуры ВС с указанием, какой процессор выполняет какую нить, какой является транзитным и какой свободен. (рис.5)

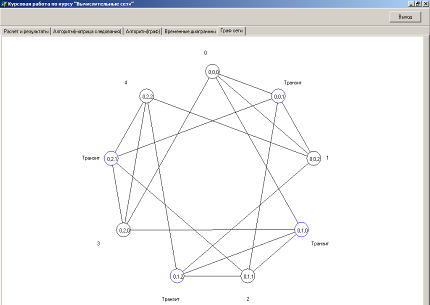


Рис.5. Окно вывода результатов

# 6. Результаты работы программы

На рисунке 6 показана сгенерированная матрица следования S.

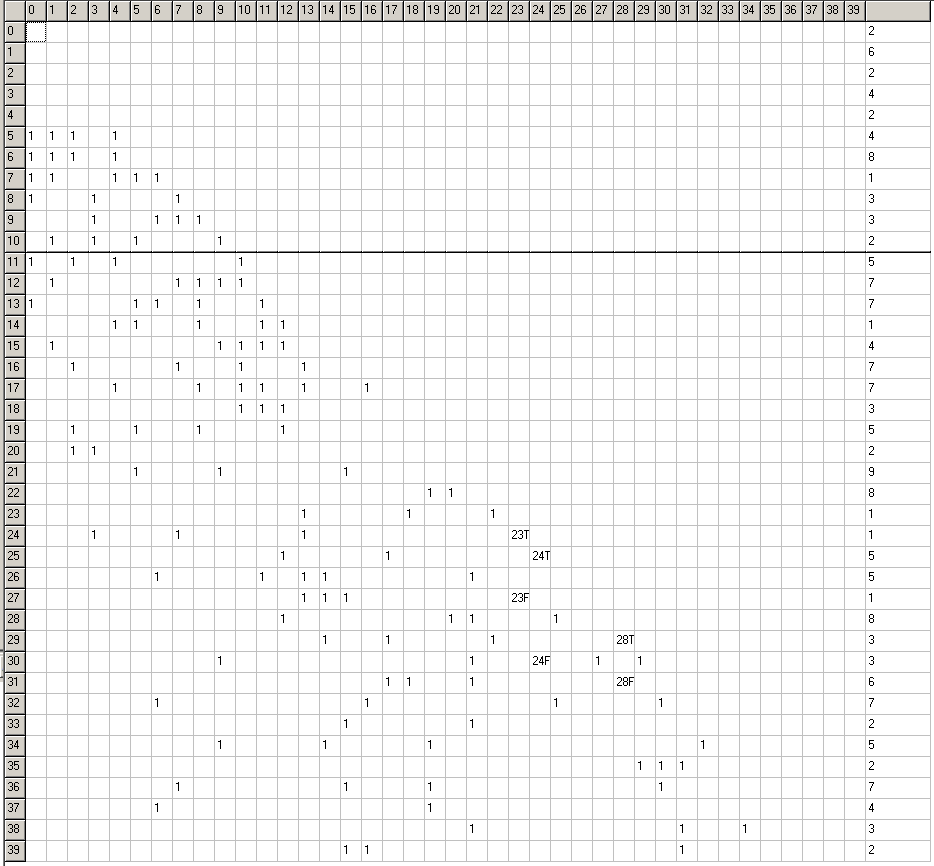


Рис.6. Сгенерированная матрица следования S

На рисунке 7 показан построенный программой по указанной в задании матрице S ИЛГ задачи.

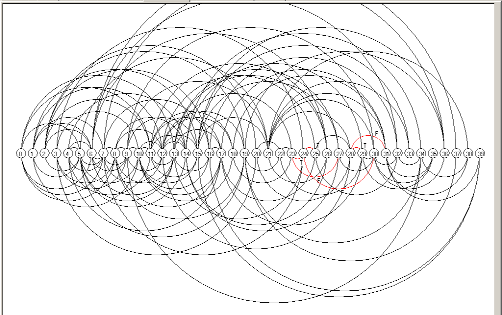


Рис.7. ИЛГ задачи.

Получим диаграммы размещения нитей на узлах ВС, задавая различные значения результатам выполнения логических операторов.

Логические связи считаются связями по данным.

Для начала рассмотрим случай, когда логические связи в ИЛГ считаются связями по данным, то есть рассмотрим ИЛГ как ИГ, заменив логические связи на связи по данным. Получаем размещение операторов по нитям (в виде перечисления и графа алгоритма), временные диаграммы, и размещение нитей по процессорам (соответственно рис 8,9,10,11).

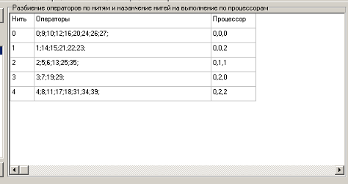


Рис.8. Размещение операторов по нитям при рассмотрении ИЛГ как ИГ (перечисление).

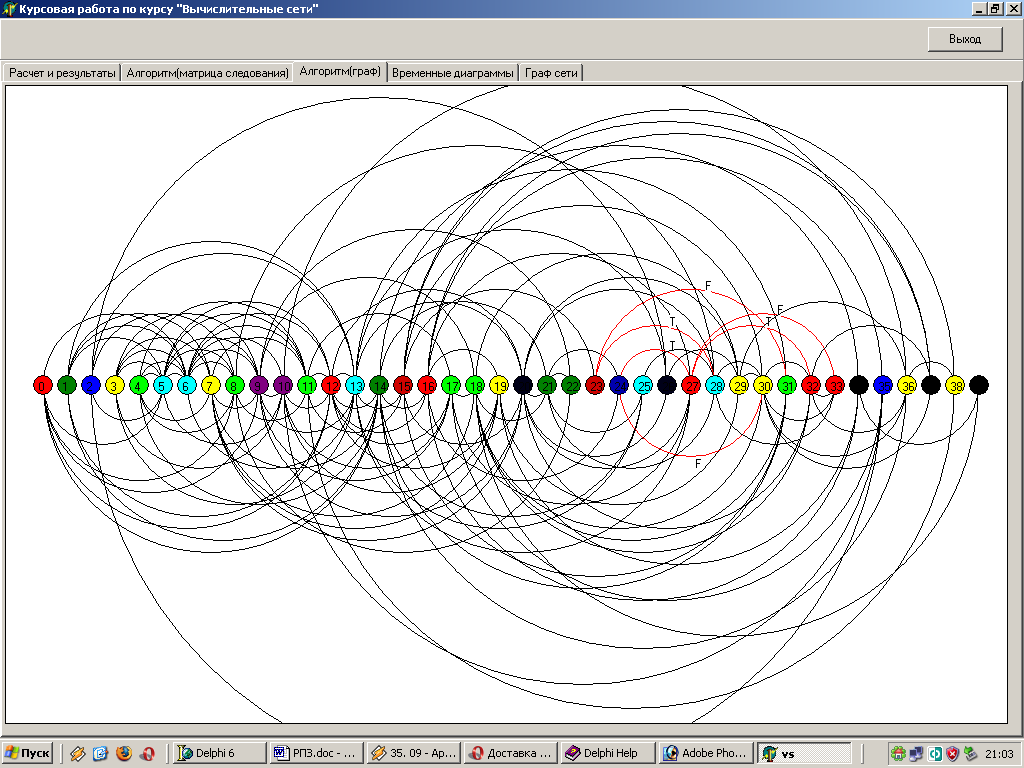


Рис.9. Размещение операторов по нитям при рассмотрении ИЛГ как ИГ (граф алгоритма).

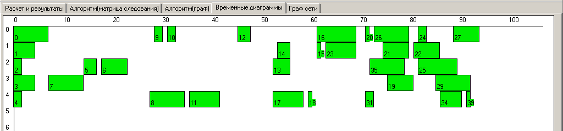


Рис.10. Временные диаграммы при рассмотрении ИЛГ как ИГ (граф алгоритма).

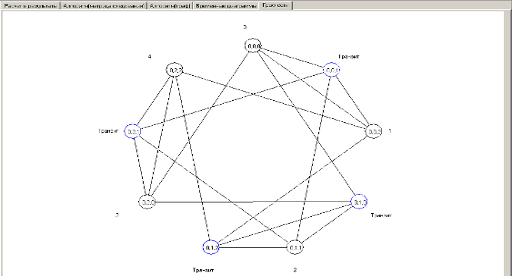


Рис.11. Распределение нитей по ВС при рассмотрении ИЛГ как ИГ (граф алгоритма)

Как видно из рисунков, ВС имеет структуру гипертора 1\*3\*3, в которой задействовано 5 процессоров, а 4 являются транзитными. Полное время решения задачи при таком подходе составляет 93 временных единицы. Неэффективное использование процессоров в ВС (соотношение рабочих процессоров и транзитных примерно 1:

1) связано с тем, что нити имеют множественную взаимную связь, что приводит к большому объему информации, передаваемых между нитями и организации транзитных процессоров.

Задействованы логические связи 23T, 24T, 27F.

Рассмотрим преобразование ИЛГ в ИГ, когда у нас известны значения логических операторов, например, 23T, 24T, 27F. На основании этих данных можно исключить из планирования те операторы, которые не будут выполняться. После этого можно осуществлять планирование выполнения алгоритма.

Результаты приведены на рисунках 12-15.

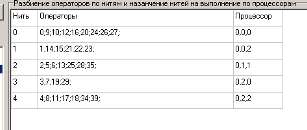


Рис.12. Размещение операторов по нитям при ИЛГ 23T, 24T, 27F.

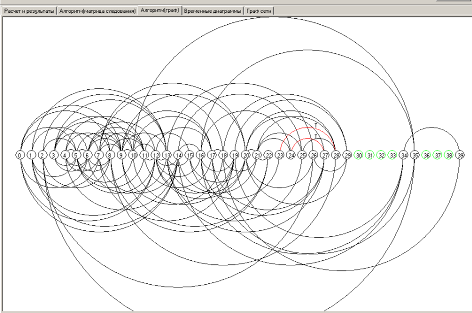


Рис.13. Размещение операторов по нитям при ИЛГ 23T, 24T, 27F.



Рис.14. Временные диаграммы при рассмотрении при ИЛГ 23T, 24T, 27F.

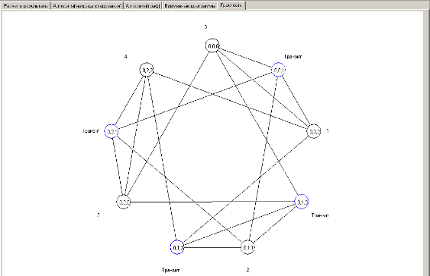


Рис.15. Распределение нитей по ВС при ИЛГ 23T, 24T, 27F.

Анализируя рис 11-15, получаем, что при преобразовании ИЛГ в ИГ часть операторов (30,31,32,33,36,37,38) была исключена из рассмотрения, что повлияло на ход выполнения планировки задания.

За счет того, что алгоритм при преобразовании был изменен, планировка вычислений изменилась и теперь длительность вычислений составляет 96 временных единиц. Это связано с тем, что при перепланировке были исключены из рассмотрения операторы, что повлияло на алгоритм планировки.

Задействованы логические связи 23F, 24F, 27F.

Рассмотрим преобразование ИЛГ в ИГ, когда у нас известны значения логических операторов, например, 23F, 24F, 27F. На основании этих данных можно исключить из планирования те операторы, которые не будут выполняться.

После этого можно осуществлять планирование выполнения алгоритма.

Результаты приведены на рисунках 16-19.

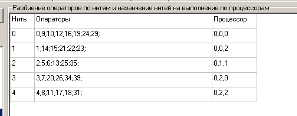


Рис.16. Размещение операторов по нитям при ИЛГ 23F, 24F, 27F.

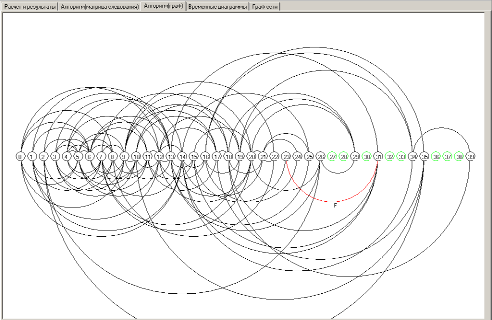


Рис.17. Размещение операторов по нитям при ИЛГ 23F, 24F, 27F.

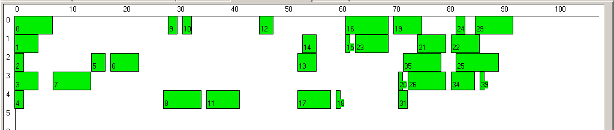


Рис.18. Временные диаграммы при рассмотрении при ИЛГ 23F, 24F, 27F.

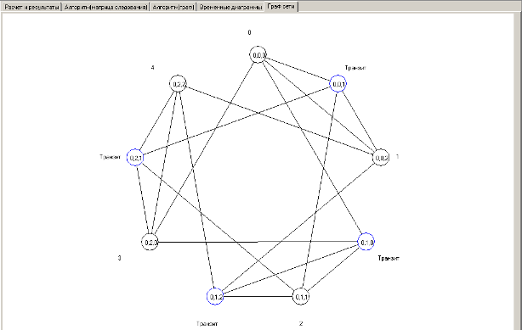


Рис. 19. Распределение нитей по ВС при ИЛГ 23F, 24F, 27F.

Анализируя рис 11-15, получаем, что при преобразовании ИЛГ в ИГ часть операторов (27,28,30,32,33,36,37,38) была исключена из рассмотрения, что повлияло на ход выполнения планировки задания. За счет того, что алгоритм при преобразовании был изменен, планировка вычислений изменилась и теперь длительность вычислений составляет 91 временную единицу. Это связано с тем, что при перепланировке были исключены из рассмотрения операторы, что повлияло на алгоритм планировки.

Выводы:

На основании полученных данных делаем вывод о том, что использование логических операторов в алгоритме может повлиять на время выполнения алгоритма как в сторону увеличения времени выполнения алгоритма, так и в сторону уменьшения времени выполнения. Поэтому при прогнозировании времени выполнения алгоритма необходимо рассмотреть все возможные комбинации результатов выполнения логических операторов. Сведем результаты в таблицу.

Таблица 1. Результаты времени выполнения алгоритма на ВС при различных значениях логических операторов.

|  |  |  |  |
| --- | --- | --- | --- |
| Тип ВС | Размерность | Значения логических операторов | Полное время решения задачи в условных единицах |
| Обобщенный гипертор | 1x3x3 | 23T,24T,27T | 96 |
| Обобщенный гипертор | 1x3x3 | 23T,24T,27F | 96 |
| Обобщенный гипертор | 1x3x3 | 23T,24F,27T | 104 |
| Обобщенный гипертор | 1x3x3 | 23T,24F,27F | 104 |
| Обобщенный гипертор | 1x3x3 | 23F,24T,27T | 94 |
| Обобщенный гипертор | 1x3x3 | 23F,24T,27F | 94 |
| Обобщенный гипертор | 1x3x3 | 23F,24F,27T | 91 |
| Обобщенный гипертор | 1x3x3 | 23F,24F,27F | 91 |

Так как нам заранее неизвестны значения логических операторов, то в качестве времени выполнения берем максимальное время выполнения алгоритма, то есть 104 временных единицы.

Алгоритм планирования нитей построен таким образом, что бы обеспечить минимизацию числа процессоров, использующихся в сети, сохраняя при этом время выполнения алгоритма минимальным. При данном ИЛГ эффективной конфигурацией является конфигурация 1\*3\*3, при других алгоритмах эффективная конфигурация сети может быть другой.

Можно заметить из показанных выше рисунков, что не все процессоры используются для вычислений. Это можно объяснить с одной стороны тем, что при большем числе процессоров увеличиваются расстояния между ними и становится невыгодно с точки зрения времени передачи использовать все процессоры, а с другой - тем, что не все вершины заданного алгоритма могут обрабатываться одновременно, так как между ними существуют зависимости по данным.

# Заключение

Разработанная в данной работе программа удовлетворяет техническому заданию. Алгоритм, лежащий в ее основе, позволяет получать оптимальное, с точки зрения времени решения задачи, распределение операторов алгоритма заданной задачи по нитям и нитей по процессорам структуры ВС типа обобщенный гипертор.

Недостатком данного алгоритма можно считать его высокую вычислительную сложность, так как фактически при получении распределения анализируются все возможные варианты размещения вершины на все процессорах. При обработке широко распространены рекурсивные алгоритмы, которые имеют экспоненциальную вычислительную сложность и потому плохо масштабируются.

Достоинством алгоритма является то, что его можно использовать для любых размерностей структуры ВС типа обобщенный гипертор.