# Содержание

[Введение 2](#_Toc282113463)

[1. Обзор существующих решений 6](#_Toc282113464)

[2. Описание заданной топологии 16](#_Toc282113465)

[3. Алгоритм планирования 18](#_Toc282113466)

[4. Описание разработанной программы 23](#_Toc282113467)

[5. Экспериментальные результаты работы программы 26](#_Toc282113468)

[6. Заключение 34](#_Toc282113469)

[Список использованной литературы 36](#_Toc282113470)

# *Введение*

В настоящее время для решения достаточно сложных задач не хватает мощности одиночных процессоров, несмотря на довольно большие величины,выражаемые в гигагерцах и сотнях мегафлопсов (мегафлопс – 1 миллионопераций с плавающей запятой в секунду). Приведем несколько направлений, где необходима высокая мощность вычислительных систем:

* моделирование сложных химических или физических процессов
* векторно-матричные вычисления
* обработка 3D графики
* хранилища информации.

Существуют два пути повышения производительности вычислительных комплексов:

* + повышение частотных возможностей элементной базы
  + увеличение количества одновременно работающих вычислителей всистеме.

Недостаток первого в том, что повышение частоты процессора ограничено размером кристалла и, судя по всему, производители уже подошли к границе использования данного метода. Так что для решения задач высокой сложности данный метод не может эффективно использоваться.

Второй путь приводит к необходимости разработки эффективного параллельного алгоритма задачи, а также задачи оптимального планирования и отображения параллельного алгоритма в структуру вычислительной системы.

По аппаратным особенностям, для решения задачи параллельного вычисления алгоритма более удобны распределенные системы.

Для эффективной работы параллельной программы на распределеннойсистеме необходим адаптивный алгоритм, которому приходится решать NP-полную задачу планирования задач на ресурсы системы. В общем видематематическое решение для данных задач на современных компьютерахможет занимать от нескольких десятков минут до нескольких месяцев.Поэтому для решения этих проблем используют эвристические подходы.

Учитывая, что исходная задача задана в виде ориентированного ациклического графа в ярусно-параллельной форме, их можноформализовать в три группы:

* + списочные
  + кластерные
  + генетические.

Списочное планирование [2], [3], [4] эффективно только для систем собщей шиной или однородных полносвязных систем. Работа алгоритма заключается в формировании спискаочереди готовых вычислительных заданий с помощью одного из эвристических методов и затем оптимального,имея в виду пересылки данных, назначения очереди вычислительных работна свободные процессоры. Две эти задачи решаются независимо одна отдругой, что приводит к неэффективности вычислений.

Кластерное планирование [5], [6], [7] в чистом виде применяется только длясистем, где нет ограничений на выделяемые ресурсы, или в масштабируемых вычислительных системах. Задания из исходного графа делятся нагруппы (кластеры). Этот процесс называется кластеризацией, и его решениев общем виде имеет экспоненциальнуювременную сложность. Задания однойгруппы (кластера) выполняются на одном процессоре.

Особенность генетического планирования [8], [9]заключаетсяввозможности решения задач планирования и назначения нераздельно одна отдругой, это приводит к уменьшениюсетевого трафика и оптимальной загрузки процессоров в системе. В результате эффективность использованияпроцессоров выше, а время решениязадачи приближается к критическому(идеальному без учета пересылок).

# *1. Обзор существующих решений*

В работах [10], [11] показано, что почти все задачи построения расписаний являются NP-полными. NP-полнота задач построения расписаний обусловила широкое применение для их решения эвристических алгоритмов, основанных на жадных стратегиях, и итерационных алгоритмов: генетические и эволюционные алгоритмы, алгоритмы имитации отжига, алгоритмы случайного поиска (ненаправленного, направленного, направленного с самообучением), алгоритмы детерминированной коррекции расписания.

Алгоритмы, основанные на жадных стратегиях, подразумевают декомпозицию задачи на подзадачи (вложении задачи в семейство более простых задач). Задача построения расписания может быть разбита на подзадачи следующими способами:

1. Подзадача заключается в распределении i-го рабочего интервала в расписание, содержащее (i-1) рабочий интервал. Для решения задачи полного составления расписания HP требуется решить N таких подзадач, где N - число рабочих интервалов в программе. В каждой подзадаче число возможных вариантов ее решения равно M0, где M0<N - число возможных мест размещения i-го рабочего интервала.

2. Строится максимально параллельное расписание - время инициализации каждого рабочего интервала определяется готовностью его к выполнению (выполнены все предшественники). Подзадача заключается в ликвидации одного процессора и распределении его процессов по оставшимся процессорам. Для решения задачи полного построения расписания HP требуется решить M′-M таких подзадач, где M - число процессоров в ВС, для которой строится расписание, M′ - число процессоров для максимально параллельного расписания. Каждая подзадача заключается в решении NS- следующих подзадач: распределение рабочего интервала, принадлежащего ликвидируемому процессору, в один из оставшихся процессоров, где NS- –числорабочихинтерваловвликвидируемомпроцессоре. Всегдасправедливо NS-<N. Число возможных вариантов решения подзадачи распределения рабочего интервала всегда меньше N.

То есть, при данных способах разбиения задачи составления расписаний на подзадачи, оптимальное решение каждой подзадачи может быть получено за полиномиальное число шагов, например, полным перебором возможных вариантов решения. Число подзадач также полиномиально. Жадные алгоритмы, использующие второй способ разбиения задачи, могут иметь более высокую вычислительную сложность по сравнению с алгоритмами, использующими первый способ (верхняя оценка сложности алгоритмов соответственно равна O(N2) и O(N3-N2⋅M)), но они применимы и в случае, когда процесс может иметь более одного рабочего интервала и на расписание наложено ограничение “всерабочиеинтервалыодногопроцессадолжныбытьназначенынаодинитотжепроцессор”.

Свойство 1. Для жадных алгоритмов решения NP-полных задач составления расписаний не существует единой для всех подзадач локальной целевой функции, приводящей к наилучшему варианту расписания, получаемого алгоритмом.

Свойство 2. Для жадных алгоритмов решения NP-полных задач составления расписаний влияние на качество расписания используемых локальных целевых функций возрастает с ростом сложности функции T=f(HP,HW) по учитываемым временным задержкам.

Свойство 3. Для жадных алгоритмов решения NP-полных задач составления расписаний критерии в локальной целевой функции, определяющие привязку рабочих интервалов к процессорам, оказывают более сильное влияние, на качество, получаемого алгоритмом расписания, чем критерии, определяющие порядок рабочих интервалов на процессорах.

В работе [10] отмечена подверженность жадных алгоритмов построения расписаний аномалиям: улучшение параметров ВС и программы, для которой составляется расписание, может приводить к увеличению времени выполнения расписания. В [12] приведены примеры, когда списочный алгоритм получает худшее по времени выполнения расписание при: 1)увеличении числа процессоров 2)уменьшении длительности выполнения рабочих интервалов 3)снятие ограничений на порядок следования рабочих интервалов 4)уменьшение длительности внутренних простоев процессоров. По крайне мере, одной из причин подверженности жадных алгоритмов аномалиями может быть свойство 1.

Итерационные алгоритмы можно разбить на два подкласса:

* алгоритмы, опирающиеся на метод проб и ошибок: генетические и эволюционные алгоритмы [8], [9], алгоритмы имитации отжига [13], алгоритмы случайного поиска (ненаправленного, направленного, направленного с самообучением) [14];
* алгоритмы детерминированной коррекции расписания.

Схематично работу этих алгоритмов для решения задачи построения расписания можно представить следующим образом:

1. Задать начальное приближение HP0, k=0.
2. Вычислить целевую функцию f(HPk,HW) и проверить выполнение ограничений (если в п.3 возможно получение ).
3. Получить HPk+1=D({HPi},{f(HPi,HW)},Prk:i∈(1,…,k)).
4. Если критерий останова не достигнут, то k=k+1 и перейти к п.2; в противном - завершить работу алгоритма.

Где, D –некотораястратегиякоррекциитекущегорасписания, Prk –параметрыстратегии (длярядастратегий, возможно, их изменение в ходе работы алгоритма).

Алгоритмы, опирающиеся на метод проб и ошибок, содержат элементы случайного выбора привязки и порядка, что дает возможность получать информацию о их влиянии на значения целевой функции и ограничений. Данная информация используется для адаптации алгоритма (в ходе его работы) к закону влияния привязки и порядка на значения целевой функции и ограничений. Кроме возможности адаптации к решаемой задаче, алгоритмы, опирающиеся на метод проб и ошибок, наиболее полно удовлетворяют требованиям, к свойствам алгоритмов построения расписания при совместном проектировании аппаратных и программных средств ВС: 1)возможностям методов строить расписания для архитектур ВС с различным уровнем детализации; 2)универсальности методов в классе допустимых архитектур ВС; 3)совместимости методов построения расписания и методов выбора оптимальных параметров архитектуры.

Данные особенности алгоритмов обеспечивают эффективность их использования на системном уровне проектирования: определение числа процессоров, “типа”каждогопроцессора, топологииихарактеристиккомпонентов (задержка, арбитражимаршрутизация) коммутационнойсреды, распределениепамяти (тип–ПЗУ/ОЗУ, объем, способдоступа–локальная/разделяемая, циклысчитывания/записи) и построение расписания выполнения прикладной программы.

Алгоритмы детерминированной коррекции расписания для своего построения требуют некоторых априорных сведений о влиянии привязки и порядка на значения целевой функции и ограничений. Это обуславливает специализацию методов на конкретный класс архитектур ВС. Однако, для уточнения расписания (локальной оптимизации расписания) можно предложить алгоритмы детерминированной коррекции расписания исправляющие “плохие”фрагментырасписанияинеиспользующиеаприорные сведения о влиянии привязки и порядка на значения целевой функции и ограничений. Если некоторый рабочий интервал находится длительное время в состояниях - ожидание данных от других рабочих интервалов или ожидание получения разделяемого ресурса, то можно попытаться уменьшить время нахождения рабочего интервала в этих состояниях путем изменения расположения в расписании рабочего интервала или его непосредственных предшественников. То есть, работа алгоритма основана на использование информации о “плохих”фрагментахрасписания, котораяможетбытьопределенаприанализединамикифункционированияВС. ТакоепостроениеалгоритмовдетерминированнойкоррекциирасписанияпозволяетустранитьихузкуюспециализациюнаконкретныйклассархитектурВС. Основнаяобласть эффективного применения алгоритмов детерминированной коррекции решений –уточнениерасписанийнауровнерегистровыхпередачприрешениизадачиуточнениядеталейреализацииВСсучетомэлементнойбазыисистемногопрограммногообеспечения.

В генетических алгоритмах система операций преобразования расписаний определена: операции скрещивания, мутации и селекции. Требуется выбрать такую форму представления расписаний, чтобы операции генетического алгоритма не приводили к получению недопустимых вариантов расписаний и любое допустимое расписание могло быть представлено в выбранной форме представления расписаний.

В алгоритмах имитации отжига, случайного поиска, детерминированной коррекции расписаний система операций преобразования расписаний вводится при разработке алгоритма. Для этих алгоритмов способ представления расписаний и система операций преобразования расписаний должными обладать следующими свойствами: применение операции должно приводить к допустимому варианту расписания и система операций должна быть функционально полной.

# *2. Описание заданной топологии*

В данном курсовомпроекте была разработана программа статического погружения графа заданий на топологию "бинарное дерево".

При использовании данной топологии система строится по схеме так называемого строго двоичного дерева, где каждый узел более высокого уровня связан с двумя узлами следующего по порядку более низкого уровня. Узел, находящийся на более высоком уровне, принято называть родительским, а два подключенных к нему нижерасположенных узла – дочерними. В свою очередь, каждый дочерний узел выступает в качестве родительского для двух узлов следующего более низкого уровня. Отметим, что каждый узел связан только с двумя дочерними и одним родительским. Пример топологии «бинарное дерево» для семи узлов представлен на рис. 2.1.



Рис. 2.1. Пример топологии «бинарное дерево» для семи узлов

Если h–высота дерева (количество уровней в древовидной системе), определяемая как [log2N], то такую систему можно охарактеризовать следующими параметрами: диаметр - D = 2(h– 1); порядок узла –d = 3; число узлов –I = N– 1; ширина бисекции –B = 1.

При больших объемах пересылок между несмежными узлами древовидная топология оказывается недостаточно эффективной, поскольку сообщения должны проходить через один или несколько промежуточных звеньев. Очевидно, что на более высоких уровнях системы вероятность затора из-за недостаточной высокой пропускной способности линий связи выше.

# *3. Алгоритм планирования*

В данномкурсовом проекте был разработан пространственный планировщик – программа, реализующая статическое планирование процессов для системы с топологией «бинарное дерево». Для этой цели был выбран генетический алгоритм.

Генетические алгоритмы являются одними из эволюционных алгоритмов, применяемых для поиска глобального экстремума функции многихпеременных. Принцип их работы основан на моделировании некоторыхмеханизмов популяционной генетики: манипулирование хромосомным набором при формировании генотипа новой биологической особи путемнаследования участков хромосомныхнаборов родителей (кроссовер) и случайное изменение генотипа, известноев природе как мутация. Другим важным механизмом, заимствованным уприроды, является процедура естественного отбора, направленная наулучшение от поколения к поколениюприспособленности членов популяциипутем большей способности к "выживанию" особей, обладающих определенными признаками.

Рассмотрим шаги генетическогоалгоритма погружения параллельнойзадачи в распределенную вычислительную систему.

1. Генерируется случайным образом N генотипов планирования графа. Информация о распределении задач по процессорам хранится в массивах. Индекс каждойячейки массива –этономервершиныв графе задачи, а значение в ячейке –т номер процессора, на котором данная подзадача будет выполняться.

2. Строится матрица маршрутизации пакетов с данными внутри системы. Определяются кратчайшие пути между каждыми двумя процессорами.

3. Осуществляется равномерныйкроссовер всех хромосом со случайным образом выбранной из списка с вероятностью 0,5.

4. Осуществляется мутация всеххромосом с вероятностью Pm = 0,05. Выполняетсяслучайное изменение значений для поиска глобального экстремума.

5. Вычисляется целевая функциядля всех генотипов:

а) сортируются все вершины по их приоритетам. Высший приоритет получает задача, которая находится на более высоком ярусе графа;

б) сортируются задачи по приоритетам пересылок данных из них.

Приоритеты пересылок соответствуютприоритетам задач, для которых нужны вычисленные данные;

в) пока все вершины не выполнены, осуществляется:

* поискневыполненныхвершин, для которых получены все данные, и процессоры которых не были занятыпри текущем событийном цикле. Времявыполнения вершинравно их весу. Задачи выполняется потактово. Маскируем процессоры так, чтобыониснова не использовались для обработки новых задач при текущем цикле исполнения;
* на остальных уже занятых процессорах уменьшаются счетчики выполнения задач;
* если задача выполнена полностью, проверяется матрица связей графа задач. Если у данной задачи есть зависимые задачи, то пересылка добавляется в очередь связи, которая выбирается из таблицы маршрутизации. Время пересылки равно весу ребра графа задач. Пересылка начинается с следующего такта и производится потактово. Для этого связь маскируется до конца такта;
* для свободных связей между процессорами проверяются их очереди. Если они не пусты, то пересылки ставятся на выполнение;
* на занятых связях между процессорами уменьшаются счетчики выполнения пересылки;
* если пересылка выполнена полностью, проверяется, сделана ли она в конечную точку. Если нет, то пересылка добавляется в очередь следующей связи, которая выбирается из таблицы маршрутизации и устанавливается флаг, чтобы не выполнить эту пересылку еще раз. Если да, то обнуляется соответствующее значение в матрице связей графа задач;
* увеличивается значение счетчика тактов на единицу;

г) время выполнения алгоритмаравно значению счетчика тактов.

6. Если это первая итерация, тосравниваются целевые функций, и M < Nлучших структур заносим в отдельныймассив. Они используются для того,чтобы решение в процессе итерацийне ухудшилось. Если это не перваяитерация, то лучшиеструктуры также участвуют в процессесравнения.

7. Уничтожаются идентичные структуры, так как процесс выбора элитных структур приводит к созданию идентичныхструктур..

8. Структуры сортируются индексной сортировкой список структур и (N–M) последних худших структур удаляются.

9. Пока не закончилось числоэпох (циклов алгоритма) или не достигнуто оптимальное для данной задачи решение, переходим на шаг 3.

# *4. Описание разработанной программы*

В разработанной программе реализована возможность ввода и редактирования погружаемого графа в графическом виде. Главное окно программы (рис. 4.1) предоставляет доступ к инструментам редактирования.

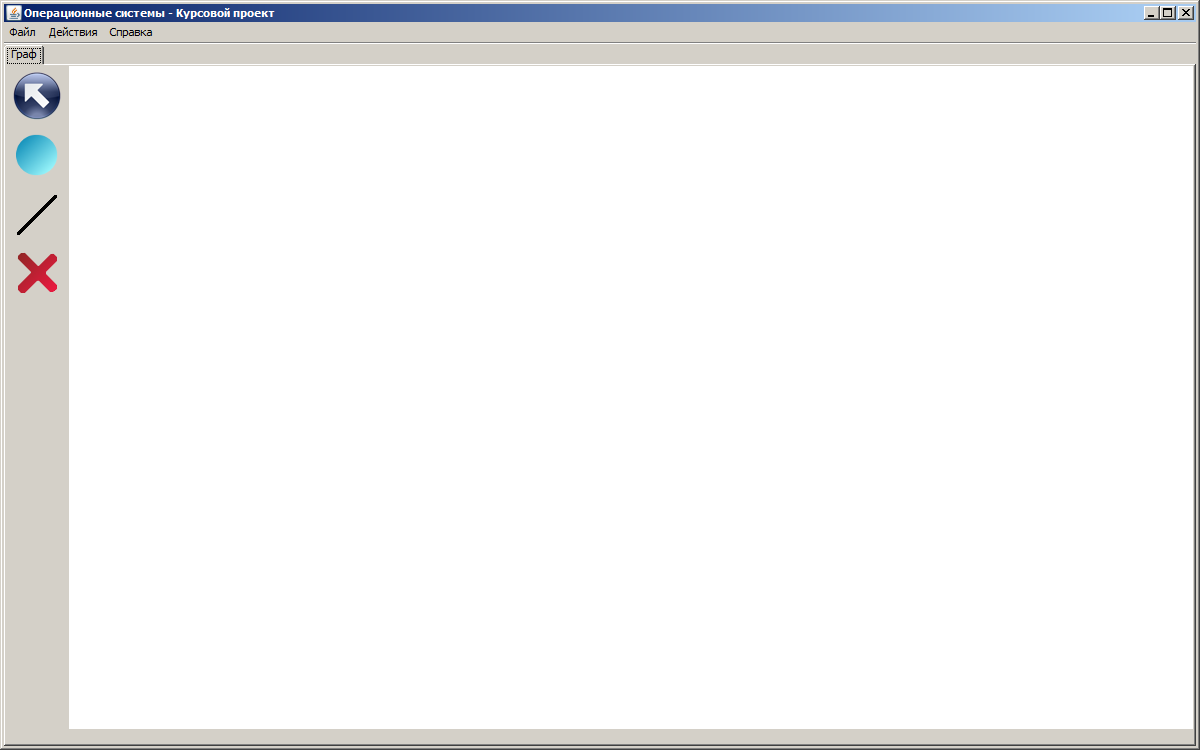


Рис. 4.1. Главное окно программы

Меню «Файл» содержит пункты: «Создать граф» (комбинация клавиш Ctrl-N), «Открыть граф…»(комбинация клавиш Ctrl-O), «Сохранить граф»(комбинация клавиш Ctrl-S), «Сохранить граф как…»(комбинация клавиш Ctrl-Shift-N), «Закрыть граф»(комбинация клавиш Ctrl-W). Они позволяют создать, закрыть, сохранить и восстановить погружаемый граф. Также меню «Файл» содержит пункт «Выход»(комбинация клавиш Ctrl-E) для закрытия программы.

Инструменты для ввода или редактирования графа находятся на панели в правой части главного окна программы. Кнопка «Перемещение вершин» позволяет изменить расположение вершины в рабочей области. Кнопки «Добавление вершины» и «Добавление ребра» предназначены для добавления в граф новой вершины и нового ребра соответственно.При добавлении вершины или ребра вызывается диалоговое окно для ввода веса вершины или ребра (рис. 4.2). Кнопка «Удалить вершину» позволяет удалять вершины из графа. Для удаления ребра необходимо вызвать контекстное меню вершины и выбрать необходимое ребро.

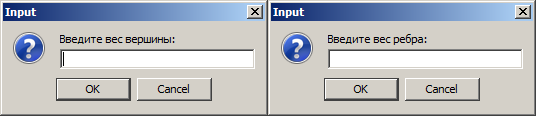


Рис. 4.2. Диалоговые окна для ввода весов новых вершины и ребра

Для запуска моделирования процесса планирования необходимо выбрать пункт «Погрузить граф…»(комбинация клавиш Ctrl-L) меню «Действия» и в диалоговом окне ввести количество процессоров в вычислительной системе. После этого в новой вкладке главного окна в виде диаграммы Ганта будут выведены результаты планирования (рис. 4.3).

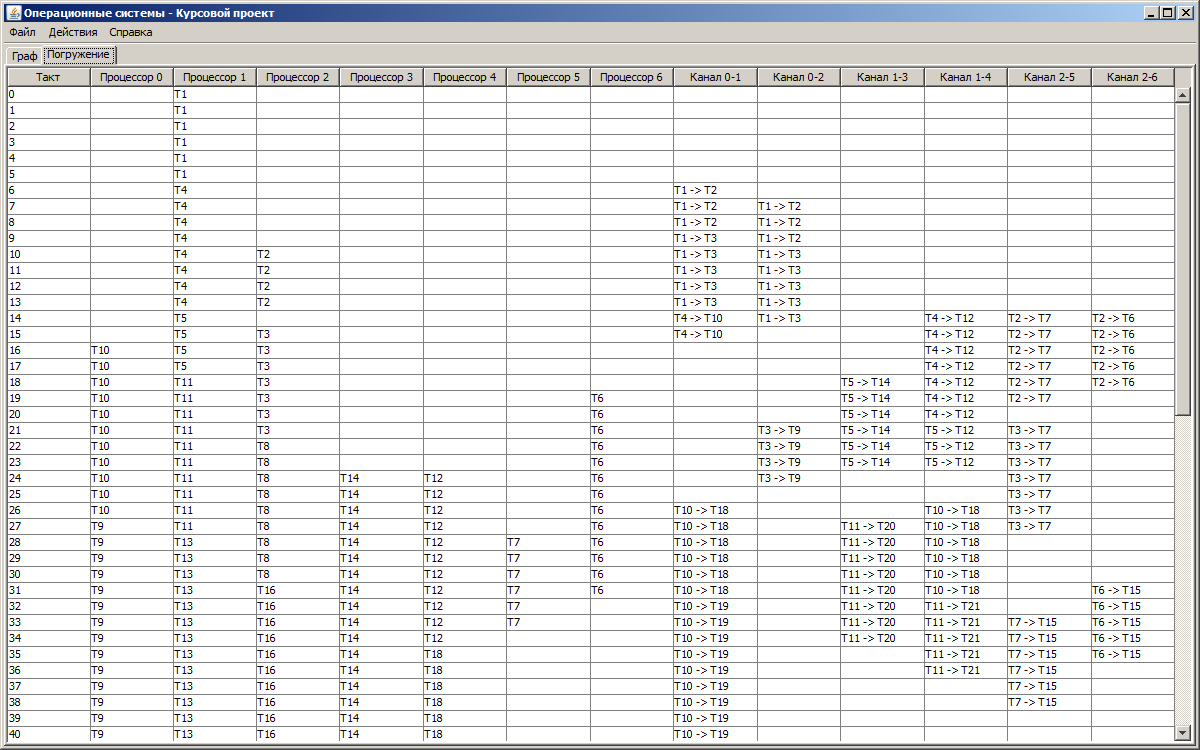


Рис. 4.3. Пример вывода результатов программы

# *5. Экспериментальные результаты работы программы*

Для эксперимента использовался граф задач, изображенный на рисунке 5.1.



Рис. 5.1. Погружаемый граф задач (24 вершины)

Построим данный граф с помощью редактора графов созданной программы. Результат изображен на рисунке 5.2.

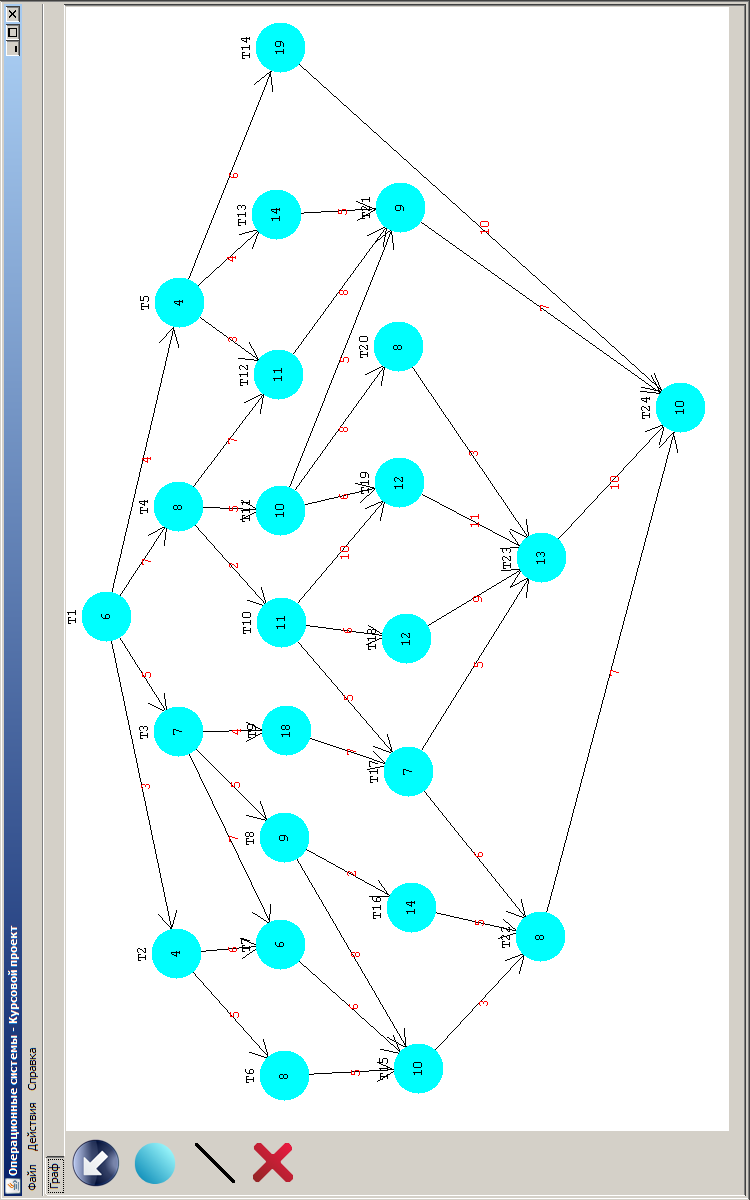


Рис. 5.2. Созданный в программе граф задач

Погрузим данный граф задач на вычислительную систему с топологией «бинарное дерево», состоящую из семи процессоров. Результаты представлены на рисунках 5.3, 5.4.

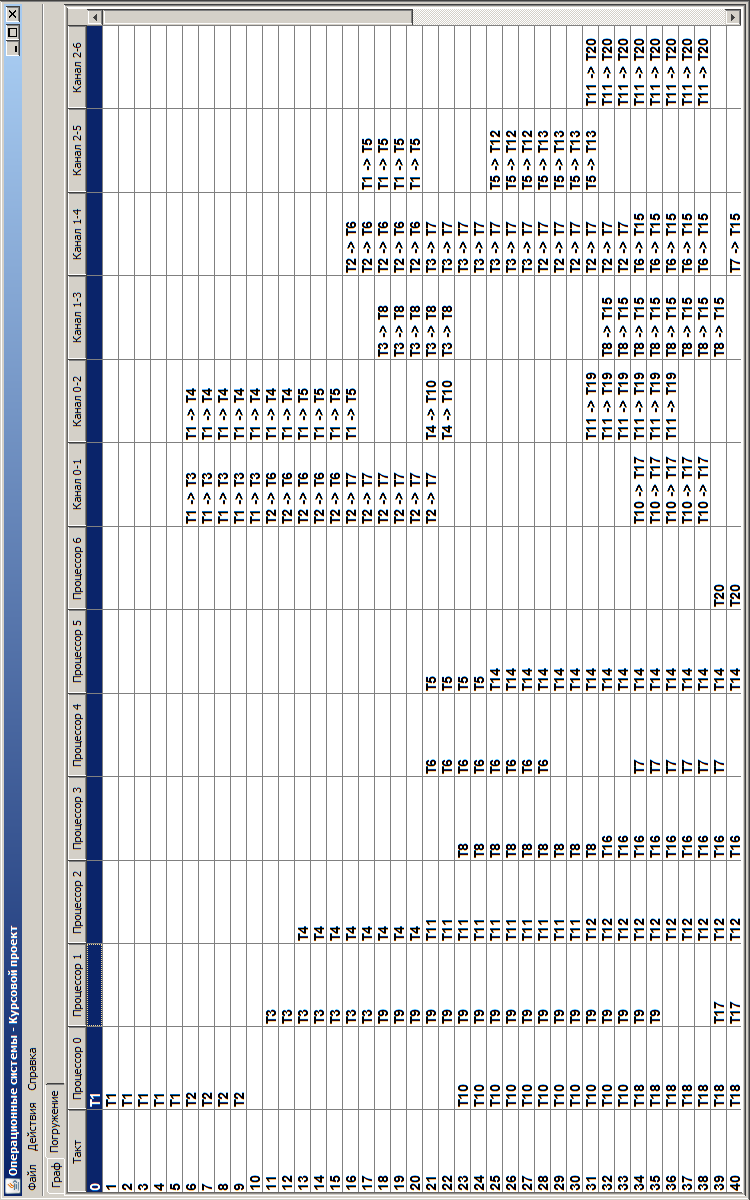


Рис. 5.3. Результаты работы программы

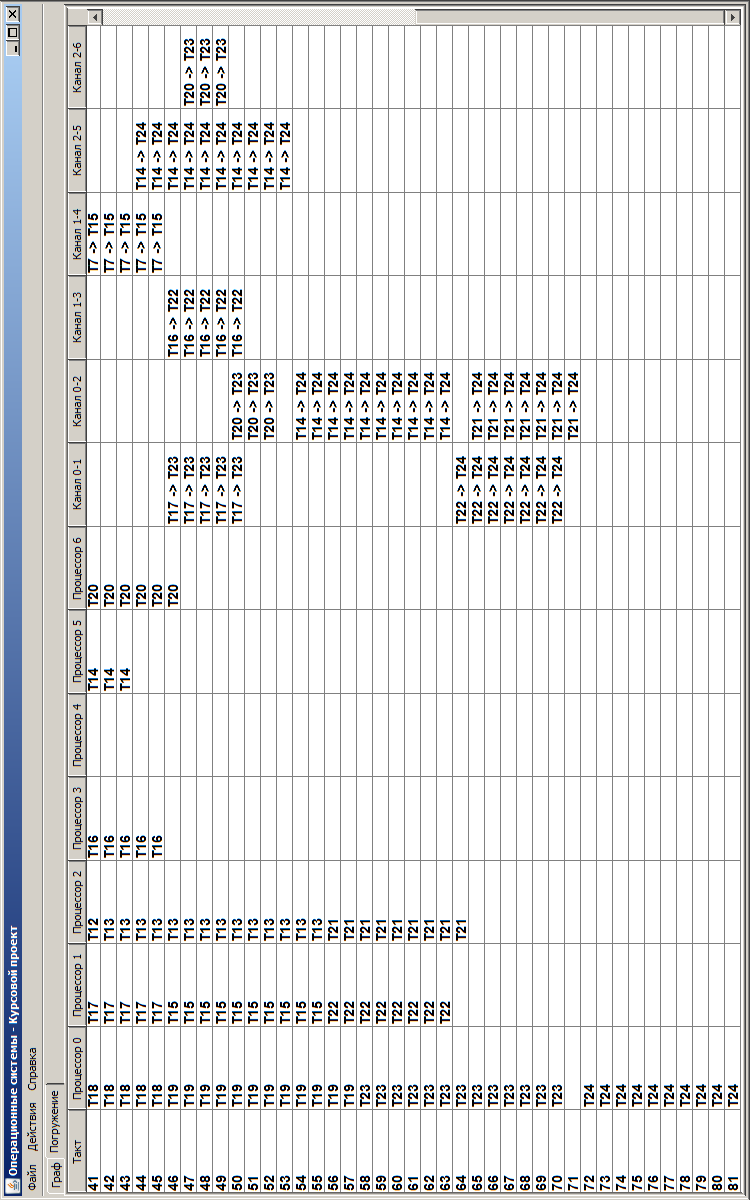


Рис. 5.4. Результаты работы программы (продолжение)

# *6. Заключение*

При выполнении курсовогопроекта мною были закреплены знания о принципах работы современных операционных систем, *изучены различные методы решения задач пространственного планирования, оптимизации, рассмотрены основные этапы планирования, а также разработан алгоритм планирования для вычислительных систем с топологией «бинарное дерево»*.

Разработанный алгоритм представляет практический интерес припостроении планировщиков многопроцессорных систем. В его разработкеиспользован современный генетический подход. Качество работы алгоритма близко к идеальному (полного перебора), причем сложность алгоритма возрастает линейно со сложностью задачи. Данный алгоритм является универсальными ориентирован для работы с распределенными и конфигурируемыми вычислительными системами.Дополнительным преимуществомданного метода является независимость от количества процессоров в вычислительной системе, так какколичество процессоров в системе является входными данными для алгоритма.

Также была построена моделирующая программа, которая демонстрирует работу разработанного алгоритма на примере загрузки пользовательского графа задач на ресурсы системы. Листинг программы представлен в Приложении А.

Выполнение данного курсовогопроекта существенно улучшило мои навыки в прикладном программировании и помогло закрепить знания, полученные по курсу «Операционные системы».

# *Список использованной литературы*

1. Конспект лекций по курсу «Операционные системы».
2. Русанова О.В., Нагорнюк В.В. Двухпроходной эвристический алгоритм планирования и отображения для систем с распределенной памятью // Вестник НТУУ "КПИ" "Информатика, управление и вычислительная техника”. – 1998. – №31. – C. 238—251.
3. Berman F. Experience with an automatic solution to the mapping problem // The Characteristics of Parallel Algorithms, MIT Press, Cambridge, MA, 1987. – P. 307—334.
4. McFarland M., Parker A., Composano R. Tutorial on high-level synthesis // Proc. 25th Design Automation Conference. ACM and IEEE. – 1988. – P. 330—336.
5. Князькова З.В., Симоненко В.П. Метод направленного поиска при статистическом планировании задач в распределенных вычислительных системах // Пробл. программирования. – 2002. –№1—2. – C. 247—252.
6. Kramer O., Muhlenbein H. Mapping strategies in message based multiprocessor systems // Lecture Notes in Computer Science. – 1987. – 158. – P. 213—225.
7. Shen H. Self-adjusting mapping: a heuristic mapping algorithm // Developing Transputer Applications, Amsterdam, IOS. – 1989. – P. 89—98.
8. Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности / Г.К. Вороновкий, К.В. Махотило, С.Н. Петрашев, С.А. Сергеев. – ОСНОВА – 1997. – 112 с.
9. Исаев С.А. Генетические алгоритмы –эволюционные методы поиска.
10. Теория расписаний и вычислительные машины/ Под ред. Э.Г.Коффмана. М.: Наука, 1984. - 334с.
11. Гэри М., Джонсон Д. Вычислительные машины и трудно решаемые задачи. - М.: Мир, 1982. - 416с.
12. Шахбадзян К.В., Тушкина Т.А. Обзор методов составления расписаний для многопроцессорных систем// Записки научных семинаров ЛОМИ –Л.: Наука, 1975., т.54 –С.229-258.
13. Уоссермен Ф. Нейрокомпьютерная техника. Теория и практика.- М.: Мир, 1992.
14. Л.А. Растригин. Статистические методы поиска.- М.: Наука, 1968.