На правах рукописи

Гончар Дмитрий Русланович

МЕТОДЫ ПЛАНИРОВАНИЯ ВЫЧИСЛЕНИЙ В САПР СИСТЕМ РЕАЛЬНОГО ВРЕМЕНИ

Специальность 05.13.11 – Математическое
и программное обеспечение вычислительных машин,
комплексов и компьютерных сетей

АВТОРЕФЕРАТ
диссертации на соискание учёной степени
кандидата технических наук

Москва – 2008

Работа выполнена в отделе Математического моделирования систем проектирования Вычислительного центра им. А.А.Дородницына РАН

Научный руководитель: кандидат физико-математических наук, доцент
Фуругян Меран Габибуллаевич

Официальные оппоненты: доктор физико-математических наук
Серебряков Владимир Алексеевич

кандидат технических наук
Кононов Дмитрий Алексеевич

Ведущая организация: Институт системных исследований РАН

Защита состоится «30» октября 2008 г. в 14 час.
на заседании диссертационного совета Д 002.017.02
в Вычислительном центре им. А.А.Дородницына РАН
по адресу: 119333, г. Москва, ул. Вавилова, 40, конференц-зал.

С диссертацией можно ознакомиться в библиотеке ВЦ РАН.

Автореферат разослан «29» сентября 2008 г.

Учёный секретарь
диссертационного совета
доктор физико-математических наук,
профессор *В.В.Рязанов*

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность работы.

Предметом исследования данной работы являются вычислительные системы *жёсткого реального времени (ВСРВ)* и вопросы автоматизации их проектирования. В таких системах части заданий назначаются директивные сроки (начала и окончания обработки заданий), не подлежащие нарушению.

ВСРВ предназначены для контроля и управления при жёстких временных ограничениях процессами в автоматизированных производствах, в производстве энергии (особенно ядерной энергетике), химической промышленности, газо- и нефтедобыче, авиационном, железнодорожном и морском транспорте, в системах управления лётными испытаниями, при управлении в чрезвычайных ситуациях. Сегодня ВСРВ всё шире применяются в управлении дорожным движением автотранспорта, экологическом и медицинском мониторинге, разнообразных системах безопасности. Таким образом, рассматриваемый класс задач имеет, помимо чисто научной, большую практическую важность.

Одновременно с увеличением масштабов и разнообразия применений ВСРВ становятся более высокими требования к их эффективности, надёжности и скорости разработки. Удовлетворить этим отчасти противоречивым требованиям представляется возможным, прежде всего, путём выхода на более высокий уровень автоматизации построения систем реального времени, созданием соответствующих эффективных, надёжных и удобных инструментов их программирования.

Одним из практически важных и весьма распространённых типов ВСРВ является ВСРВ с периодически поступающей на обработку входной информацией. Если при этом возможна приемлемая точность оценки времени на обработку входной информации, возникает возможность разбиения процесса работы ВСРВ на две стадии – предварительную и основную.

На предварительной стадии и не в режиме реального времени первоначально описывается (например, средствами специально построенного для этих целей формального языка) состав и периодичность поступающей информации, соответствующие модули-обработчики, последовательность обработки данных, директивные сроки обработки тех или иных данных и допустимость прерывания этой обработки и т.д. Затем (с учётом необходимой синхронизации процессов обработки, предотвращения программных тупиков и учёта иных особенностей программируемых ВСРВ) автоматически рассчитывается допустимое расписание работы ВСРВ (либо делается вывод о невозможности существования такого расписания), составляется с использованием средств автоматизации программирования программа выполнения режима реального времени, обеспечивается создание нужного количества копий программных модулей обработчиков входных и промежуточных данных ВСРВ (с учётом возможности одновременной обработки нескольких поколений входных данных), средств связи между этими модулями и ОС реального

времени и т.п., а на основном этапе производится собственно выполнение полученной ВСРВ в режиме реального времени.

Именно такой подход, позволяющий не только качественно повысить эффективность, скорость и надёжность разработки (настройки, усовершенствования) ВСРВ, но и существенно расширить возможный диапазон и сложность решаемых задач (особенно при управлении быстропротекающими процессами) был предложен в ВЦ АН СССР лауреатом Премии СМ СССР, к.ф.-м.н. Борисом Григорьевичем Сушковым (1941-1997) и во многом осуществлён под его руководством для ЕС ЭВМ.

Уже в 90-е годы была понята актуальность и важность разработки подобной системы и для однопроцессорных персональных ЭВМ. Такая система САПР «СРВ-Конструктор» была создана с непосредственным существенным участием автора диссертации и подробно представляется в данной работе.

В настоящее время автором продолжаются дальнейшие разработки, направленные на развитие системы для многопроцессорных вычислительных комплексов, в частности, на разработку более эффективных алгоритмов задачи построения расписания для ряда важных частных случаев, часто встречающихся в работе подобных систем.

Отметим, что совершенствование технологии и конструкции вычислительных средств (на основе кластеров, многоядерных процессоров и т.п.) в последние годы сделали многопроцессорные вычислительные комплексы качественно доступнее и поэтому обоснованность их применения в упомянутых областях ещё более возрастает.

Вышеперечисленные обстоятельства обусловливают актуальность исследований в указанной области.

В математическом плане центральной задачей автоматизированного построения ВСРВ, как правило, является задача составления расписаний. Многие варианты задач составления многопроцессорного расписания являются NP-трудными. Причём любая практическая реализация составления многопроцессорного расписания – это всегда своеобразный компромисс между результатом и вычислительной сложностью. Поэтому вопрос составления более эффективных эвристических алгоритмов, в том числе для конкретных видов задач составления расписаний, является в настоящее время достаточно актуальным для рассматриваемой предметной области.

Создание и использование ВСРВ стало актуальной задачей при появлении достаточно надёжной и мощной вычислительной базы, что, как известно, произошло к 70-х годам XX в. С этого времени изучению математических вопросов теории расписаний и её приложений исследователи уделяют повышенное внимание.

В России, а до того в СССР, задачей построения расписаний в системах реального времени занимались Танаев В.С., Барский А.Б., Головкин Б.А., Сушков Б.Г., Шкурба В.В., Гордон В.С., Костенко В.А., Лазарев А.А., Мищенко А.В., Португал В.М., Сигал И.Х., Шафранский В.С. и др.

Среди зарубежных учёных следует отметить Бернса (Burns), Брукера (Bruker), Гонзалеса (Gonzales), Греневельта (Groenevelt), Гэри М. (Garey),

Дертузоса (Dertouzos), Джонсона Д.(Johnson), Конвея Р.В. (Conway), Кормена Т. (Cormen), Коффмана (Koffman), Лью(Liu), Лейланда (Layland), Лейзерсона Ч.(Leiserson), Максвелла В.Л.(Maxwell), Мартеля (Martel), Миллера Л.В.(Miller), Мока (Mok), Одсли (Audsley), Рамамритама (Ramamritham), Ривеста Т. (Rivest), Ричардсона (Richardson), Сахни (Sahni), Станковича (Stankovic), Ульмана Дж.(Ullman), Федергрюна (Federgruen) и др.

Считаю своим долгом выразить благодарность Б.Г.Сушкову, Ю.А.Флёрову, М.Г.Фуругяну, О.Л.Кондратьеву, О.С.Федько и С.Н.Мирошнику за помощь в работе и обсуждение полученных результатов.

Цели работы.

Основной целью диссертационной работы является разработка программного комплекса инструментальной САПР «СРВ-Конструктор» для персональных электронно-вычислительных машин, а также новых методов составления расписаний, предназначенных для функционирования на многопроцессорной ВС. Эти методы можно включить в состав программных средств, предназначенных для осуществления планирования вычислений на вычислительных системах, в том числе системах жёсткого реального времени.

Для достижения поставленной цели:

– создан программный комплекс, реализующий инструментальную САПР «СРВ-Конструктор»;

– с целью дальнейшего совершенствования указанной САПР для многопроцессорных вычислительных комплексов разработаны и реализованы новые эвристические алгоритмы решения задачи построения оптимального по быстродействию расписания без прерываний.

СРВ-Конструктор состоит из:

Блока синтаксического и семантического анализа, осуществляющего синтаксический и семантический анализ конструкций программы реального времени (РВ-программы), описывающей на формальном языке необходимый порядок выполнения прикладных программ пользователя, генерирующего таблицы данных для работы последующих блоков и вычисляющего размеры буферов обмена данными между программными модулями.

Блока генерации сетевой модели и расписаний**,** строящего математическую модель вычислений, выполняемых в реальном времени, определяющего возможность построения допустимого расписания выполнения прикладных модулей и само расписание, если оно существует, и выполняющего некоторые другие вспомогательные функции построения инструментальной САПР ВСРВ.

Блока генерации кода, формирующего на языке Си и записывающего в текущий каталог исходные тексты получившейся программы, а также создающего ряд вспомогательных файлов для последующих определённых действий пользователя, компиляции и редактирования связей.

Управляющего монитора на основе многозадачной оболочки реального времени CTask-RT, обеспечивающего работу в реальном времени прикладной программы пользователя, сгенерированной на этапе предварительной обработки посредством САПР ВСРВ.

Научная новизна работы.

Научная новизна диссертационной работы заключается в построении инструментального программного комплекса, дающего новые качественные возможности при автоматизации построения систем реального времени с периодическим поступлением входной информации, а также разработке алгоритмов построения расписаний работ для многопроцессорной вычислительной системы.

В процессе исследования автором было выполнено следующее:

– разработана и программно реализована инструментальная система САПР «СРВ-Конструктор», включающая язык реального времени, блоки синтаксического и семантического анализа, блок генерации сетевой модели и расписаний, блок генерации кода и управляющий монитор;

– разработаны и программно реализованы алгоритмы решения минимаксной задачи составления расписаний без прерываний с использованием различных правил предпочтения при выборе определения допустимого расписания;

– на основе многочисленных вычислительных экспериментов получены апостериорные оценки точности разработанных алгоритмов.

Методы исследований.

Методологическую и теоретическую основу исследования составили методы разработки систем реального времени, теории расписаний, комбинаторной оптимизации и теории графов.

Практическая ценность работы.

Основные результаты исследования относятся к созданию нового инструментального программного комплекса САПР «СРВ-Конструктор» для однопроцессорных ПЭВМ и разработке ряда перспективных эвристических алгоритмов оптимизации планирования вычислений в системах реального времени для многопроцессорных вычислительных комплексов для дальнейшего развития системы САПР «СРВ-Конструктор».

Разрабатываемые алгоритмы и тесты программ могут быть применены при построении и сопровождении автоматизированных систем управления энергетическими установками, химическими производствами, системами транспорта, мониторинга экологических, медицинских и экономических явлений.

А то, что в последние годы начато подлинно массовое производство и применение многопроцессорных технических средств (от вычислительных

кластеров до многоядерных процессоров для ПЭВМ), делает применение таких САПР как «СРВ-Конструктор» всё более доступным и целесообразным.

Апробация работы.

Результаты диссертации и материалы исследований докладывались и обсуждались на:

- III научной школе "Автоматизация создания математического обеспечения и архитектуры систем реального времени" (Саратовский ф-л Института машиноведения РАН, Саратов, 1992);
- межд. конф. "Проблемы управления в чрезвычайных ситуациях" (М., ИПУ РАН, 1992);
- межд. конф. "Проблемы регионального и муниципального управления" (М.: РГГУ, 1999);
- IX и X межд. конф. "Проблемы управления безопасностью сложных систем" (М., ИПУ РАН, 2001 и 2002);
- III межд. конф. по исследованию операций (М., ВЦ РАН, 2001);
- науч. конф. "Математические модели сложных систем и междисциплинарные исследования" (М., ВЦ РАН, 2002);
- XLV и XLIX науч. конф. Московского физико-технического института (ГУ), (М.-Долгопрудный, 2002, 2006);
- II Всеросс. науч. конф. «Методы и средства обработки информации» (М.: МГУ, 2005);
- научных семинарах сектора проектирования систем реального времени Вычислительного центра им. А.А. Дородницына РАН;
- научных семинарах кафедры математических основ управления Московского физико-технического института (ГУ).

Публикации. По материалам диссертации опубликовано 19 печатных работ, в том числе одна в журнале, входящем в список изданий, рекомендованных ВАК. Список работ приведён в конце автореферата.

Структура и объём работы.

Диссертация состоит из введения, семи глав, заключения, приложений и списка литературы. Общий объём работы составляет 139 страниц.

СОДЕРЖАНИЕ РАБОТЫ

Во введении говориться о важности и актуальности построения вычислительных систем реального времени в целом и рассмотрена общая схема включения ЭВМ в контур контроля процессов, происходящих в некотором реальном физическом объекте, основные потоки информации и состав аппаратных средств типичной многоканальной системы сбора данных и управления процессами (см. рис. 1).

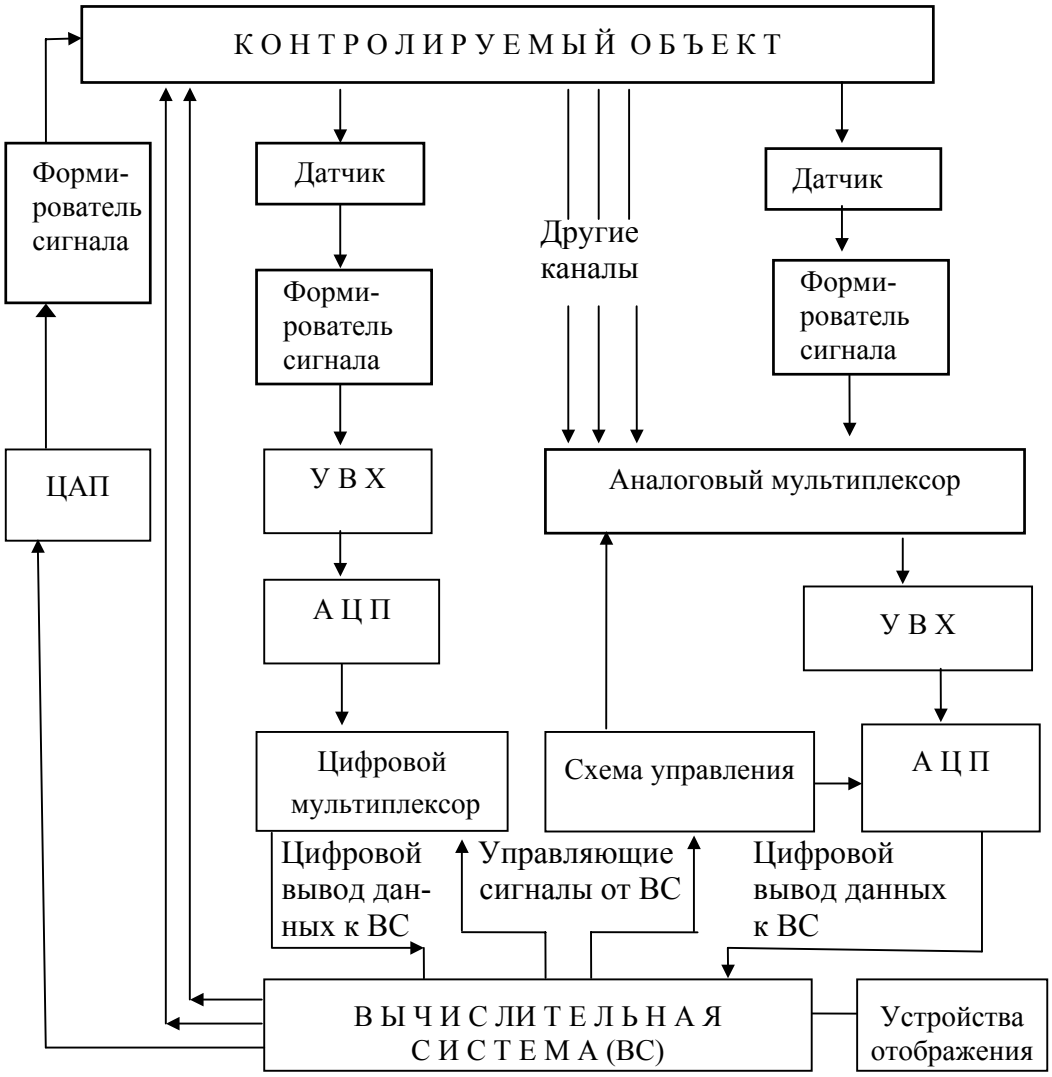

Рис. 1. **Схема использования ЭВМ для управления физическими процессами**. *Потоки информации и аппаратные средства типичной многоканальной системы сбора данных и управления представляются на различных устройствах отображения (дисплеях и т.п.) и, кроме того, если необходимо, вырабатываются управляющие воздействия, которые после соответствующих преобразований поступают к объекту, управляя его функционированием. АЦП – аналогово-цифровой преобразователь. ЦАП – цифро-аналоговый преобразователь. УВХ – устройства выборки-хранения аналогового сигнала.*

Первая глава диссертации посвящается общему описанию САПР систем реального времени «СРВ-Конструктор», как основного проектируемого инструментального средства. Описывается состав и структура САПР ВСРВ в целом.

Существует два основных подхода к созданию САПР ВСРВ. Первый связан с расширением существующих языков новыми операторами, позволяющими пользователю описывать параллельные вычисления и их синхронизацию. При этом вся работа по разделению времени для однопроцессорных систем ложится на пользователя.

При создании ВСРВ «СРВ-Конструктор» был выбран другой, более перспективный подход, при котором вся работа по разделению времени и оптимизации вычислений выполняется транслятором. Для осуществления такого подхода необходимо предварительно построить математическую модель ВСРВ, с помощью которой могут быть решены такие задачи, как разбиение программ на параллельные процессы, составление допустимого расписания выполнения процессов, синхронизация вычислений, динамическое распределение памяти.

Для генерации прикладной ВСРВ пользователю необходимо иметь прикладные модули, написанные на языках программирования, например *Си, Фортран, Паскаль* или *Ассемблер*, и написать на специальном языке реального времени задание на обработку информации в реальном времени – РВ-программу, где задаётся порядок обработки прикладными модулями входных данных и отображения результатов счёта по отношению к периоду поступления кадров данных в систему. Если данный порядок обработки может быть соблюдён, система обеспечит реализацию заказанной обработки. В противном случае выдаётся сообщение о невозможности вести указанную обработку.

В отличие от систем потоковой обработки данных предусмотрена возможность работы прикладных модулей с несколькими поколениями данных. Система автоматически обеспечивает хранение этих данных в специальных буферах нужное время. При возникновении внештатной ситуации с управляемым объектом, либо для изменения порядка работы программы реального времени, оператором может быть использована предусмотренная возможность быстрой реакции на поступление апериодической информации. Также предусмотрено выполнение прикладных модулей в фоновом режиме.

Вся остальная работа по генерации прикладной ВСРВ будет выполнена автоматически с помощью разработанной САПР ВСРВ. При этом будут решены следующие проблемы:

1) проблема синхронизации параллельных процессов в ВСРВ;

2) проблема тупиков при распределении ресурсов вычислительной системы (процессоров, каналов, памяти, периферийных устройств, а также информационных ресурсов);

3) проблема завершения тех или иных вычислений к определённым моментам времени или к моментам определённых событий в системе, указанных пользователем в РВ-программе (соблюдение директивных сроков);

4) проблема сохранения и обновления необходимых наборов поколений данных, как поступающих в ВСРВ извне, так и являющихся результатами вычислений с помощью РВ-программы;

5) проблема обнаружения в режиме реального времени ошибок и сбоев и организация соответствующих реакций на них.

Решение этих задач обеспечивает надёжность программного обеспечения ВСРВ.

Разработанная система наиболее эффективна при составлении программ для детерминированных ВСРВ с циклическим поступлением инфор-

мации от контролируемого объекта. Детерминированность означает здесь следующее:

1) входная информация поступает на входные регистры системы с известным периодом *T*, т.е. в моменты времени 0 ,*T,* 2*T*, ... ;

2) входная информация может быть объединена в кадры, имеющие постоянную часть (т.е. поступающую в каждый дискретный момент *0, T,* 2*T,*...) и несколько переменных частей, каждая из которых поступает в систему реже, с большим периодом, кратным *T*. Количество переменных частей кадра произвольно;

3) контролируемый процесс состоит из конечного числа детерминированных участков. Для каждого такого участка заранее известно, какие данные необходимо получать от объекта и передавать к нему, определены форматы входных кадров, а также известно, какую обработку необходимо производить, т.е. задан список программных модулей, осуществляющих нужные вычисления. Иными словами, для каждого участка контролируемого процесса известен *режим обработки*;

4) для каждого программного модуля известна оценка времени его исполнения, а также потребности в других ресурсах системы;

5) в процессе выполнения каждого режима обработки может быть вычислен момент смены текущего режима и однозначно определён следующий режим обработки;

6) задан начальный (стартовый) режим обработки (например, режим ожидания первого кадра информации от контролируемого объекта).

При разработке САПР ВСРВ для IBM PC были реализованы

1) язык реального времени;

2) транслятор с этого языка, включающий блоки синтаксического и семантического анализа, построения сетевой модели вычислений и нахождения допустимого расписания выполнения вычислений, автоматической генерации объектного кода программы реального времени;

3) управляющий монитор, контролирующий вычисления в режиме реального времени.

Во второй главе приводится подробное описание входного языка реального времени для ВСРВ «СРВ-Конструктор».

Основным простейшим объектом языка является модуль – обычный для многих языков программирования оператор процедуры. Все процедуры, участвующие в образовании модулей, составляются заранее и помещаются в системную библиотеку процедур вместе с необходимыми спецификациями формальных параметров. Имеется один специальный тип модуля, введённый для облегчения составления РВ программ – это РВ модуль. Он по описанию и смыслу ничем не отличается от обычного модуля. Но кроме описания он ещё имеет тело, внутри которого указаны вызовы других, в том числе и РВ модулей. Для РВ модулей имеется ограничение – они не должны содержать рекурсивных вызовов. Фактические параметры любого модуля могут быть РВ-параметрами, константами, простыми переменными, идентификаторами с индексами и т.д. РВ-параметры имеют следующий вид:

<РВ-параметр> ⇒ (<имя РВ-параметра>; <поставщик>;
<поколение>)

Компонента <поставщик> однозначно указывает программный модуль, в котором вычисляется запрашиваемое модулем-потребителем значение параметра с данным именем. Из дальнейшего описания языка будет видно, что вызовы модулей могут повторяться в программе, а сами модули могут входить в более сложные объекты языка. Поэтому эта компонента, в свою очередь, состоит из нескольких полей. При отсутствии имени поставщика считается, что этот РВ-параметр присутствует во входных данных.

Компонента <поколение> определяет момент времени, в который необходимо взять требуемое значение параметра. В этом разделе может быть указано время, кратное периодичности поступления информации от указанного поставщика данных. Системой будет передано потребителю последнее имеющееся на заданный момент времени значение этого параметра.

Другие виды фактических параметров аналогичны параметрам процедур известных языков программирования.

Из модулей может быть составлен следующий по иерархии сложности объект языка – цикл реального времени:

<цикл РВ> ⇒ <имя цикла РВ>, <период>, <фаза>,
<тело цикла РВ>

Компонента <период> задаёт интервал времени, через который будет повторяться выполнение модулей, указанных в теле цикла РВ. Задание периода аналогично заданию времени для РВ-параметра. Если в программе реального времени указано несколько источников данных или в качестве базового периода берётся период другого цикла РВ, то требуется явное указание его имени перед значением периода. Кроме этого период может быть задан в абсолютных величинах времени: секундах, миллисекундах. Параметр <фаза> указывает сдвиг начала работы РВ-цикла относительно начала работы программы. Правила задания параметра <фаза> такие же, как и для <периода>. Фаза не может быть задана больше периода, указанного для данного РВ-цикла. Если фаза явно не указана, то она равна нулю. <Тело цикла РВ> – это список модулей, РВ-параметры которых могут поставляться из блоков входной информации, из модулей других циклов РВ и из модулей данного цикла.

Поименованная совокупность РВ циклов образует безусловное задание (простое задание).

<безусловное задание> ⇒ **smplpgm** <имя задания>;
head <список модулей> **end**;
tail <список модулей> **end**;
inherit <список безусловных заданий>;
<список циклов РВ> **end**

Безусловное задание может быть снабжено конструкциями предварительной и заключительной части. Они служат для проведения набора специ-

фических действий перед началом работы и соответственно после конца работы простого задания.

Для облегчения программирования на языке реального времени, в новое простое задание допускается вставить несколько других простых заданий (конструкция **inherit**). В этом случае не придётся дублировать исходный текст ранее определённого простого задания.

Основную часть простого задания составляют РВ-циклы. Все циклы реального времени, входящие в одно простое задание, должны рассматриваться как выполняющиеся параллельно во времени. Если модули некоторого цикла требуют на свой вход данные, поставляемые другим циклом, то необходимая задержка организуется системой.

В ходе обработки эксперимента может потребоваться изменить режим обработки. Такая возможность предусматривается в языке реального времени введением условного задания.

<условное задание> → **condpgm** <имя задания>;
cvector <вектор условий>;
switchtable <таблица переключений>
end;
<флаги и очереди>
<список простых заданий>;
end

В разделе <вектор условий> определяется список булевых переменных, от значения которых будет зависеть порядок исполнения программы реального времени. Здесь же указываются начальные значения компонент вектора. Компоненты вектора условий могут быть использованы в качестве входных и выходных параметров РВ-модулей. В разделе <таблица переключений> задаются правила переключения между простыми и условными заданиями, в зависимости от конкретных значений вектора условий. Таблица переключений представляется в виде списка значений вектора условий и имени простого или условного задания, на которое требуется переключиться. Для таблицы переключений имеется два ограничения:

1) В таблице переключений обязана присутствовать строка, соответствующая начальному значению вектора условий.

2) Таблица переключений должна однозначно определять на какое задание должно происходить переключение.

Наконец, РВ-программа представляет собой совокупность условных заданий, все таблицы условий которой не содержат ссылок на неописанные задания.

<программа РВ> ⇒ **rtpgm** <имя программы>;
<описание констант, модулей, переменных, входных данных>;
<условные задания> **end**;

В третьей главе приводится описание системы автоматизированного синтеза модели выполнения программ в реальном времени и генератора сетевой модели и расписаний.

Реализация алгоритмов композиции подмоделей в комплексную модель оформлена в виде системы автоматизированного синтеза модели выполнения программ в реальном времени, которая состоит из следующих блоков: генератор рабочих таблиц (ГРТ), генератор сетевых моделей и расписаний (ГСМР) и генератор кодов (ГК).

На вход блока Генератор рабочих таблиц подаётся исходный текст РВ-программы, которая написана на языке сборки композиции моделей и в которой пользователь описывает, как следует объединить имеющиеся подмодели в комплексную модель. Каждая подмодель должна быть оформлена в виде прикладного модуля.

Генератор рабочих таблиц выполняет следующие функции:

1. Осуществляет синтаксический анализ конструкций РВ-программы.
2. Выдаёт сообщения о наличии ошибок в РВ-программе.
3. Генерирует таблицы данных для работы последующих блоков.
4. Вычисляет размеры буферов обмена данными программных модулей.

Генератор сетевых моделей и расписаний (ГСМР) выполняет следующие функции:

1. Строит математическую модель вычислений в виде графа, в котором вершины соответствуют прикладным модулям, а дуги определяют частичный порядок их активизации.
2. Определяет директивные интервалы выполнения прикладных модулей.
3. Определяет, существует ли допустимое расписание выполнения прикладных модулей, и строит его, если оно существует.
4. Определяет необходимое количество копий для каждого прикладного модуля.
5. Вычисляет размеры буферов для входных параметров прикладных модулей.
6. Назначает стеки прикладным модулям для работы с данными.

В главе подробно описываются математические алгоритмы, лежащие в его основе ГСМР, а именно:

- алгоритм построения сети модулей;
- алгоритмы вычисления и коррекции директивных интервалов;
- алгоритмы проверки существования допустимого расписания выполнения модулей и его построения (при положительном ответе на первый вопрос);
- алгоритм определения размеров буферов для входных параметров и др.

Блоки предварительной обработки написаны на языке Си и отлажены на многочисленных примерах, в частности, на примере решения задачи регрессионного анализа.

В **главе 4** описывается **управляющая программа** САПР «СРВ-Конструктор».

Работу в реальном времени прикладной программы пользователя, сгенерированной на этапе предварительной обработки, должна обеспечить *управляющая программа* (УП). Основные функции УП описаны ниже.

Хотя операционная система MS-DOS в чистом виде не позволяет организовать многозадачность, но, в отличие от известных систем, предназначенных для этих целей (Unix, OS/2 и т.п.), она обладает такими несомненными достоинствами, как надёжная файловая система и простота использования. Существует множество компиляторов для языков высокого уровня, рассчитанных на работу в среде MS-DOS. В силу этих соображений, выбор был остановлен на MS-DOS и сосредоточены усилия на создании приемлемой многозадачной оболочки реального времени.

Управляющую программу удалось реализовать в виде многозадачной надстройки над операционной системой MS-DOS версии 3.30 и выше на основе многозадачной оболочки реального времени CTask-RT, прототипом которой послужил распространяемый бесплатно пакет Т.Вагнера. CTask-RT позволяет создать программную среду, использующую многозадачные элементы BIOS IBM PC/AT, обойти барьеры нереентерабельности MS-DOS и даёт возможность работать совместно с резидентными (TSR) программами и под управлением оболочки MS-Windows.

Следует заметить, что написанная Управляющая программа достаточно мобильна, т.е. может быть перенесена на другие операционные системы и машины, поскольку почти 90% кода программы написано на языке Си.

Управляющая программа является составной частью исполняемого EXE-модуля РВ-программы и **выполняет следующие функции**:

- приём, хранение и обработка поступающей извне информации (кадров данных);
- реакция на внешние апериодические сигналы;
- исполнение команд, вводимых с клавиатуры консоли;
- переключение между *условными* и *простыми* заданиями в соответствии со значениями системного вектора условий;
- запуск процессора согласно директивным срокам и приоритетам;
- обмен данными между процессами;
- действия по завершению работы процесса.

Структура управляющей программы. Логически управляющую программу можно условно разделить на следующие части:

- *интерпретатор команд;*
- *диспетчер (ядро);*
- *монитор данных;*
- *драйверы внешних устройств.*

Интерпретатор команд обеспечивает интерфейс между оператором и системой РВ. Он представляет собой систему меню, работа с которой возможна как с помощью клавиатуры, так и посредством «мыши».

Примерная схема сеанса работы интерпретатора управляющей программы может быть следующей. После запуска EXE-модуля РВ-программы начинает работать интерпретатор команд. На экране появляется главное меню. Выбор из меню осуществляется либо нажатием «горячей» клавиши подсвеченной буквы, либо мышью. Сначала доступны для выбора поля **StartUp** и **Quit**, что соответствует запуску и выходу из системы. После выбора **StartUp** появляется подменю с полями **Overview, Help, Info, Install & Run**, посредством которых можно получить дополнительную информацию по системе или начать работу.

Функция *диспетчера* – активизация и запуск на счёт процессов в соответствии с приоритетами, которые определяются блоком ГСМР при составлении расписания.

Множество процессов РВ-программы состоит из процессов реакции на внешнее апериодическое сообщение (по одному на каждое УЗ), основных процессов ПЗ и фоновых процессов УЗ.

Каждый процесс состоит из блока управления процесса (шапки) и тела процесса (Е-модуля). Е-модули создаются блоком генератора кодов согласно математической модели. В блоке управления содержится вся информация о параметрах процесса: типы параметров вызова, их имена и, возможно, параметры глубины по времени выборки данных.

При запуске процесса блок управления получает управление, генерирует запрос *монитору данных*, получает требуемые параметры, после чего запускает процесс. После окончания выполнения процесса управление передаётся в блок управления, который вновь генерирует запрос монитору данных на передачу данных от процесса к потребителю. После передачи данных процесс завершает работу.

Монитор данных обеспечивает приём и хранение кадров данных, поступающих в систему извне с помощью драйверов внешних устройств, поставку этих параметров процессам, обмен данными между процессами, изменение системного вектора условий и флагов готовности/ ожидания фоновых процессов.

Драйверы внешних устройств позволяют принимать по прерываниям или/ и посредством последовательного опроса приходящие извне кадры данных и поставляют их монитору. В настоящее время реализован следующий стандартный набор драйверов внешних устройств: драйверы клавиатуры, принтера и последовательного порта (через который по умолчанию поступают внешние данные). Могут обслуживаться несколько печатающих устройств и коммуникационных портов. Для подключения нестандартного оборудования нужно лишь добавить соответствующий драйвер, что обеспечивает достаточную гибкость системы.

В **главе 5** описывается генерация кода, сборка и запуск программного комплекса «СРВ-Конструктор».

Ранее уже упоминалось, что различают подготовительный этап использования «СРВ-Конструктора», по окончании которого генерируется готовая к

выполнению прикладная РВ-программы, и этап работы в самом режиме реального времени полученной программы.

Для запуска подготовительного этапа от пользователя требуются:

– прикладные модули, написанные на языках программирования C, FORTRAN, PASCAL, ASSEMBLER (в формате компиляторов фирмы Microsoft v. 6.0;

– задание на обработку информации в реальном времени, написанное на входном языке СРВ-Конструктора.

В связи с различием описания данных в разных языках программирования, использующие дисковые файлы прикладные модули должны быть написаны на языке Си.

Общая схема предварительного этапа показана на рис. 2..

Предварительный этап работы системы «СРВ-Конструктор» включает последовательный вызов следующих обрабатывающих программных комплексов:

– компилятора для трансляции исходного текста РВ-программы; формат обращения: lang <имя_РВ-программы.rts>;

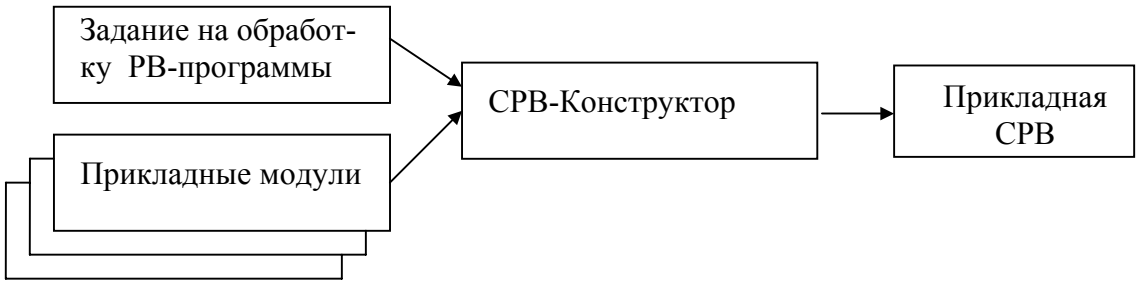

Рис.2. Последовательность проектирования ВСРВ с помощью СРВ-Конструктора.

– программы вывода диагностики и сообщений об ошибках при компиляции, помещаемых компилятором в файл <имя_РВ-программы.lst> (предусмотрен анализ более 70 типов ошибок); формат обращения: spl;

– программы генерации сетевых моделей и расписаний; формат обращения: _gsm_gr;

– программы генерации кодов; формат обращения: _cd_gnr.

Далее осуществляется трансляция полученных модулей с помощью компилятора фирмы Microsoft.C (6.0) и сборка готового модуля прикладной программы реального времени с помощью стандартного компоновщика связей фирмы Microsoft link. Файл с расширением <имя_РВ-программы.lrf>, используемый при работе данного компоновщика (включает перечень предназначенных для сборки файлов), создаётся автоматически.

Для выполнения расписания может понадобиться создание копий некоторых прикладных модулей пользователя. Список пар имён таких модулей (имеющееся имя и то, которое необходимо создать) выдаётся в файл «_cg_user.lst». Для успешной работы компилятора необходим также ряд вспомогательных файлов.

Блок генерации кода выполняет следующие функции:

1. Формирует на языке Си и записывает в текущий каталог исходные тексты получившейся программы.

2. Создаёт ряд вспомогательных файлов для последующих определённых действий пользователя, компиляции и редактирования связей.

В шестой главе даётся формальная постановка задачи построения оптимального по быстродействию расписания выполнения M независимых заданий на N процессорах в системе реального времени. Рассматривается случай выполнения заданий без прерываний. Производительность каждого процессора считается в одном случае одинаковой, затем рассматривается более общий случай с различной производительностью процессоров.

Постановка задачи для случая процессоров одинаковой производительности. Имеется M независимых заданий и N идентичных процессоров, на каждом из которых может выполняться одновременно не более одного задания. В фиксированный момент времени каждое задание выполняется не более чем одним процессором. Задания выполняются без прерываний и переключений с одного процессора на другой. Для каждого задания i известна длительность t_i его выполнения. Необходимо определить такое распределение заданий по процессорам, чтобы время выполнения всей совокупности заданий (от начала первого до завершения последнего) было минимальным.

Эту задачу можно описать следующим образом: $\overline{T} \to \min$,

$$\begin{cases} \sum_{i=1}^{M} t_i x_{ij} \le \overline{T}, & j = \overline{1,N}, \\ \sum_{j=1}^{N} x_{ij} = 1, & i = \overline{1,M}, \\ x_{ij} = \{0,1\}, & i = \overline{1,M}, \quad j = \overline{1,N}, \end{cases} \qquad (1)$$

где $\overline{T}$ - время выполнения всей совокупности заданий; $x_{ij} = 1$, если i-е задание распределено на j-й процессор, $x_{ij} = 0$ в противном случае.

Постановка задачи для случая процессоров различной производительности. Имеется M независимых заданий и N процессоров, которые могут отличаться производительностью. На каждом процессоре одновременно может выполняться не более одного задания. В фиксированный момент времени каждое задание выполняется не более чем одним процессором. Задания выполняются без прерываний и переключений с одного процессора на другой. Объём работы по выполнению задания i равен Q_i. Производительность процессора j равна S_j. Таким образом, если i-е задание выполняется процессором j в течение интервала t_{ij}, то $Q_i = S_j \cdot t_{ij}$.

Будем считать, что задания упорядочены по не возрастанию объёмов их работ: $Q_1 \ge Q_2 \ge \ldots \ge Q_M$, а процессоры упорядочены по не возрастанию их производительностей: $S_1 \ge S_2 \ge \ldots \ge S_N$.

Требуется определить такое распределение заданий по процессорам, чтобы время выполнения всей совокупности заданий (от начала первого до завершения последнего) было минимальным.

Эту задачу можно записать следующим образом: $\overline{T} \to \min$,

$$\begin{cases} \sum_{i=1}^{M}\left(\frac{Q_i}{S_j}\right)x_{ij} \le \overline{T}, & j=\overline{1,N}, \\ \sum_{j=1}^{N} x_{ij} = 1, & i=\overline{1,M}, \\ x_{ij} = \{0,1\}, \quad i=\overline{1,M}, & j=\overline{1,N}, \end{cases} \qquad (2),$$

где $\overline{T}$ - время выполнения всей совокупности заданий; $x_{ij}=1,$ если i-е задание распределено на j-й процессор, $x_{ij}=0$ в противном случае.

Обе сформулированные выше задачи, как показано в литературе, являются *NP-трудными* в сильном смысле.

Приводится обзор существующих методов решения поставленных задачи. Основное внимание уделено следующим известным методам:

- алгоритмы случайного поиска (ненаправленного, направленного, с самообучением);
- алгоритмы детерминированной коррекции расписаний;
- алгоритмы имитации отжига;
- генетические алгоритмы;
- алгоритмы агрегирования.

На основе обзора сделан вывод, что с учётом особенностей задач целесообразно совершенствовать методики построения расписаний по всем указанным направлениям. В силу научных интересов автор сосредоточил внимание на новых эвристических алгоритмах.

В **седьмой главе** подробно изложены предлагаемые эвристические алгоритмы.

Решение задачи 1 (для случая **одинаковых процессоров**).

При работе эвристического алгоритма вычисляется идеальная оценка $t^* = (\sum_{i=1}^{M} t_i)/N$ времени завершения всей совокупности заданий, а также её калиброванное значение $\bar{t}$. Под калибровкой понимается увеличение величины t^* на некоторое число процентов, причём алгоритм запускается с разными значениями калибровки и выбирается лучшее достигнутое для рассматриваемого набора входных данных решение.

Эвристический алгоритм 1.

а) Отсортировать задания по не возрастанию длительностей, т.е. будет выполняться соотношение $t_i \ge t_{i+1},\ i=1,\ldots,M-1$.

б) Вычислить величины t^* и $\bar{t}$.

в) Распределять на каждый процессор j, начиная с первого, нераспределенные ранее задания до тех пор, пока суммарная длительность заданий на каждом процессоре не будет превышать величины $\bar{t}$. При этом сначала максимально нагружается текущий выбранный процессор, а лишь затем происходит переход к загрузке заданий следующего. Если в какой-то момент при попытке распределить очередное задание на процессор происходит превышение величины $\bar{t}$, то сначала производится попытка распределения на тот же процессор оставшихся нераспределённых заданий вплоть до наименее трудоёмкого (также с не превышением величины $\bar{t}$) и только потом – переход на загрузку следующего процессора (для каждого $j = 1, ..., N$ определять $x_{ij} = 1$ до тех пор, пока $\sum_{i=1}^{M} x_{ij} t_i < \bar{t}$ для данного j).

г) Определить номер k^* процессора, для которого суммарная длительность заданий, назначенных на него, минимальна, т.е.

$$k^* = \arg\left(\min_{k=1,N} \sum_{i=1}^{M} x_{ij} t_i\right) \qquad (2)$$

д) Распределить на этот процессор самое длинное из ранее не распределённых заданий.

е) Повторять выполнение пунктов (г) и (д) до тех пор, пока не все задания распределены.

Результаты ряда экспериментов с эвристическим алгоритмом 1 приведены на рис. 3.

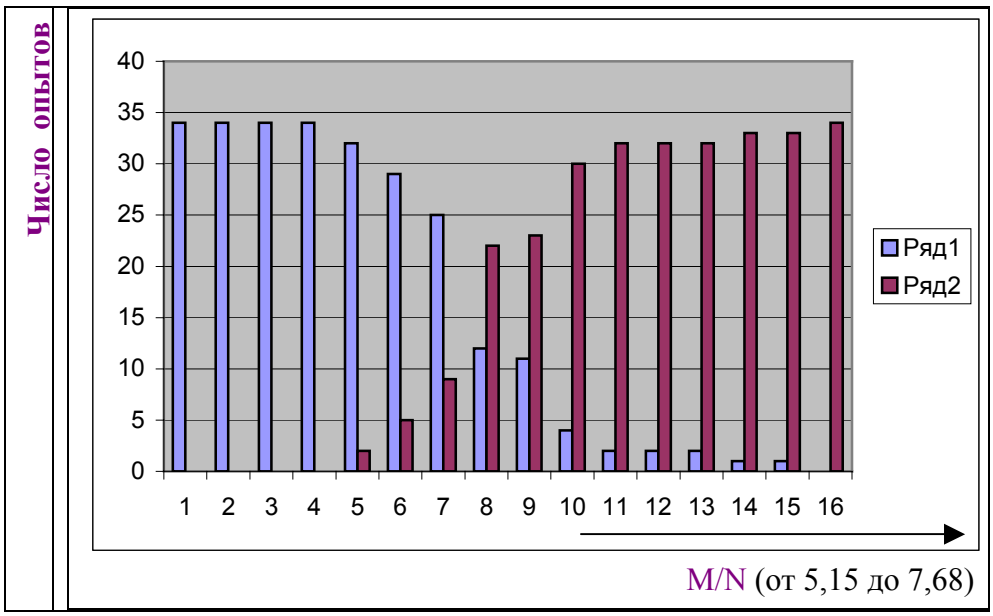

Рис. 3. Результаты ряда экспериментов с эвристическим алгоритмом 1

Ряд 1 (■) – эвристический алгоритм 1 лучше, чем «жадный».

Ряд 2 (■) – результаты совпадают.

Контрольные алгоритмы. Для сравнительного контроля качества получаемых предложенными алгоритмами решений использован известный «жадный» алгоритм распределения заданий. Как уже упоминалось, он состо-

ит в том, что распределение осуществляется, начиная с самого трудоёмкого из необработанных заданий, на наименее загруженный на момент распределения процессор.

Для оценки качества получаемых решений при малых размерностях задачи использовался и метод полного перебора вариантов распределения заданий по процессорам. Этот алгоритм хорошо известен и не требуется его описание в подробностях. Оценка его трудоёмкости составляет величину порядка $O(N^M)$.

Таким образом, оценки трудоёмкостей обсуждаемых эвристических алгоритмов определяются трудоёмкостью предварительной сортировки и, в зависимости от выбранного метода, составляют либо $O(M^2)$, либо $O(M \log M)$, соответственно. Оба вида сортировки используются в связи с тем, что при малых M (M ≤ 14), трудоёмкость сортировки методом пузырька оказывается ниже, чем сортировки разделением-слиянием**.**

Решение задачи 2 (для случая **различных процессоров**).

Описание жадного алгоритма.

1. Пусть L_j – длина временного интервала загруженности j-го процессора.
2. Положить $L_j = 0 \ \forall\, j = \overline{1, N}$.
3. Назначить первое задание на первый процессор и $L_1 := Q_1 / S_1$.
4. Для каждого $i = \overline{2, M}$ выполнять шаги 5-7.
5. Положить $R_j = \max(L_1, L_2, \ldots, L_{j-1}, (L_j + Q_i / S_j),\ L_{j+1}, \ldots, L_N)$ для $\forall\, j = \overline{1, N}$
6. Вычислить $R_{j_0} = \min\limits_{j=\overline{1,N}} R_j$.
7. Назначить задание i на процессор j_0: $L_{j_0} := L_{j_0} + Q_i / S_{j_0}$.

Описание эвристического алгоритма 2.

При работе эвристического алгоритма вычисляется нижняя оценка $t^* = (\sum_{i=1}^{M} Q_i) / \sum_{j=1}^{N} S_j$ времени завершения всей совокупности заданий, а также её калиброванное значение $\bar{t}$. Под калибровкой понимается увеличение величины t^* на некоторое значение, причём алгоритм запускается с разными значениями калибровки и выбирается лучшее достигнутое для данного набора входных данных решение. В качестве значения верхней оценки T_G длины расписания используется время выполнения всей совокупности заданий, полученное «жадным» алгоритмом для соответствующих данных.

Алгоритм распределения заданий.

1) Вычислить величины $t^*,\ T_G$. Пусть K – заданное число (значение параметра). Алгоритм запускается для следующих значений $\bar{t}$:

$$\bar{t} = t^* + \left(\frac{t_G - t^*}{K} \right) \cdot h,\ h = \overline{0, K}.$$

2) Распределять на каждый процессор j, начиная с первого, нераспределённые ранее задания до тех пор, пока суммарная длительность заданий на каждом процессоре не будет превышать величины $\bar{t}$. При этом сначала максимально загружается текущий выбранный процессор, а лишь затем происходит переход к загрузке заданий следующего. Если в какой-то момент при попытке распределить очередное задание на процессор происходит превышение величины $\bar{t}$, то сначала производится попытка распределения на тот же процессор оставшихся нераспределённых заданий вплоть до наименее трудоёмкого (также с не превышением величины $\bar{t}$) и только потом – переход на загрузку следующего процессора (для каждого $j = 1, ..., N$ определять $x_{ij} = 1$ до тех пор, пока $\sum_{i=1}^{M} x_{ij} \frac{Q_i}{S_j} < \bar{t}$ для данного j). Для каждого j вычислить величину L_j.

3) Назначить оставшиеся ранее нераспределённые задания согласно пп. 3-5 «жадного» алгоритма.

Для предложенных алгоритмов даётся оценка точности их вычислений. Приводятся результаты серии расчётов по данным алгоритмам и сравнительный анализ последних.

Результаты ряда экспериментов с эвристическим алгоритмом 2 приведены на рис. 4.

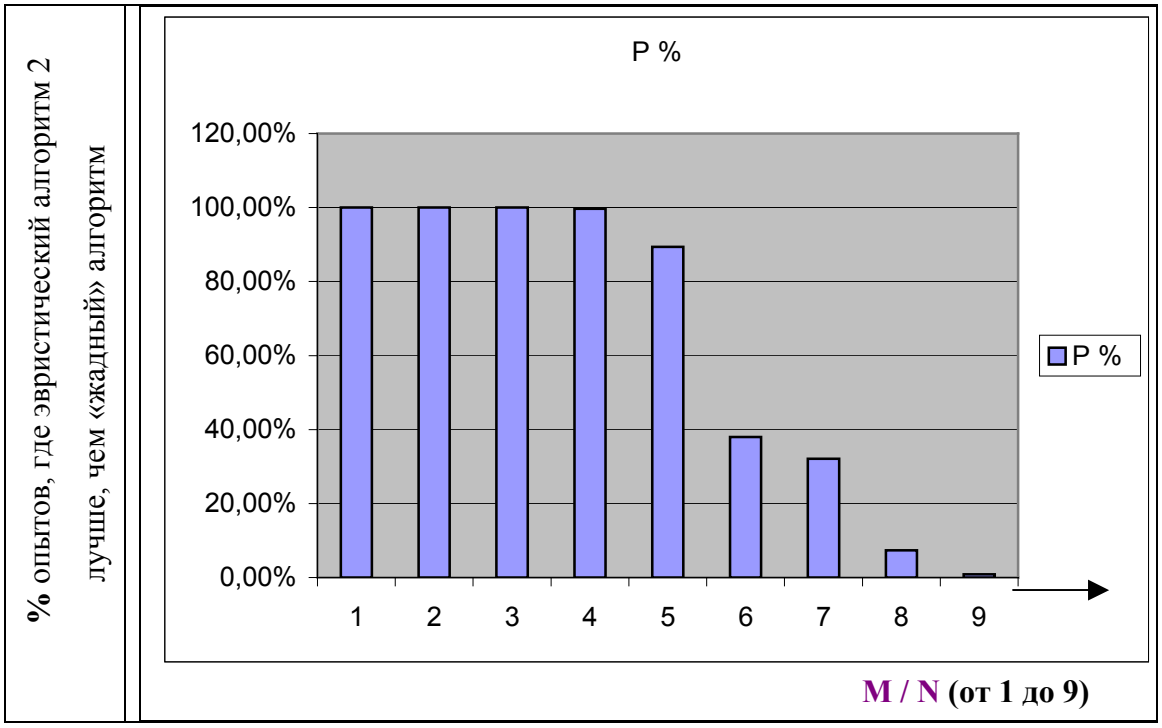

Рис. 4. *Результаты ряда экспериментов с эвристическим алгоритмом 2*

$$\max \frac{P_{j_1}}{P_{j_2}} = 2 \quad \forall \; j_1, j_2 \le N.$$

Проведённые эксперименты показали, что с используемыми механизмами калибровки (а) мультиоценочный алгоритм целесообразно применять при соотношении числа заданий к числу процессоров, не превосходящем 7. (б) в дополнение к мультиоценочному алгоритму следует использовать также и «жадный» алгоритм, поскольку оба алгоритма работают достаточно быстро и при разных данных могут давать различные по точности результаты.

В Приложении 1 приведён пример применения САПР ВСРВ для решения в реальном времени задачи многомерной линейной регрессии.

В Приложении 2 - тексты программной реализации предлагаемых эвристических алгоритмов для многопроцессорного варианта САПР «СРВ-Конструктор».

В заключении сформулированы основные теоретические и практические результаты диссертационного исследования.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ ДИССЕРТАЦИОННОЙ РАБОТЫ

Основные результаты диссертационной работы заключаются в следующем:

1. Для однопроцессорных персональных ЭВМ типа IBM PC создан инструментальный программный комплекс автоматизации проектирования систем реального времени «СРВ-Конструктор», позволяющий в автоматизированном режиме проектировать вычислительные системы реального времени.

В рамках создания данной САПР решены следующие задачи:

- формального описания на языке высокого уровня программы выполнения прикладных модулей пользователя с учётом состава и периодов поступления информации в каждый модуль;
- трансляции этой программы (используется разработанная в ВЦ РАН система построения компиляторов «Супер»);
- автоматического определения наличия допустимого расписания выполнения поставленной задачи;
- генерации кодов и сборки исполнимых модулей готовой к выполнению ВСРВ.

2. С целью дальнейшего совершенствования указанной САПР для многопроцессорных вычислительных комплексов разработаны и реализованы новые эвристические алгоритмы решения задачи построения оптимального по быстродействию расписания без прерываний. Данные алгоритмы основаны на мультиоценке и калибровке получаемых результатов и выборе наилучшего среди сгенерированных результатов.

Основные результаты диссертации опубликованы в работах:

1. *Сушков Б.Г., Фуругян М.Г., Гончар Д.Р., Кондратьев О.Л.* Методология и программные средства мониторинга чрезвычайных ситуаций. //Тез. докл. конф. "Проблемы управления в чрезвычайных ситуациях", Москва, ИПУ РАН, 1992, С. 62-63.

2. *Сушков Б.Г., Фуругян М.Г., Гончар Д.Р., и др*. САПР систем реального времени на базе IBM PC "СРВ-конструктор" //Тез.докл. III научной школы "Автоматизация создания мат. обеспечения и архитектуры систем реального времени". Саратовский филиал Института машиноведения РАН, Саратов, 1992. С. 3-4.

3. *Гончар Д.Р.,* САПР систем реального времени для IBM PC (сборник трудов). М., ВЦ РАН, 1993. С. 83-87.

4. *Сушков Б.Г., Фуругян М.Г., Гончар Д.Р. и др*. САПР вычислительных систем реального времени для IBM PC. // Тез. межд. конф. "Проблемы регионального и муниципального управления". Москва: РГГУ, 1999, 1 с.

5. *Сушков Б.Г., Фуругян М.Г., Гончар Д.Р. и др*. Система автоматизации программирования вычислительных систем реального времени. - В сб. "Теория и реализация вычислительных систем реального времени". М.: ВЦ РАН, 1999, 8 с.

6. *Сушков Б.Г.,* Белый Д.В., *Фуругян М.Г., Гончар Д.Р.,* Кондратьев О.Л. *и др*. Автоматизация программирования вычислительных систем реального времени. //В сб.: Моделирование обработки информации и процессов управления. М.: МФТИ, 1999, 10 с.

7. *Gonchar D., Fourougyan M.* The decision of one problem of distribution of resources at presence of uncertain factors. Тез. 3-й Моск. межд. конф. по исследованию операций. Москва, ВЦ РАН, 2001 г., 1 с.

8. *Гончар Д.Р., Фуругян М.Г.* Автоматизация проектирования систем реального времени для испытаний и мониторинга сложных технических объектов. - Материалы 9-й межд. конф. "Проблемы управления безопасностью сложных систем", М., ИПУ РАН, 2001, 4 с.

9. *Гончар Д.Р., Фуругян М.Г.* Алгоритмы анализа и синтеза многопроцессорных систем реального времени. Тез. конф. "Математические модели сложных систем и междисциплинарные исследования". М: ВЦ РАН, 2002. 1 с.

10. *Гончар Д.Р.,* Эвристические алгоритмы распределения заданий в многопроцессорной системе реального времени. Труды XLV науч. конф. МФТИ (ГУ), 2002 г., часть VII, М.-Долгопрудный: МФТИ, 2002. 1 с.

11. *Гончар Д.Р., Гуз Д.С.*, *Красовский Д.В., Фуругян М.Г.* Эффективные алгоритмы планирования вычислений в многопроцессорных системах. // Проблемы управления безопасностью сложных систем: Труды 10-й международной конференции. Книга 2. /ИПУ РАН. – М., 2002 – С. 207-211.

12. *Gonchar D.* Multiboumds Heuristic Algorithm for Design of Scheduling M Tasks on N Equal Processors.// Труды 4-й Моск. межд. конф. по исследованию операций (ORM 2004). Москва: (ВМиК МГУ), 2004 г. С. 97-98.

13. *Гончар Д.Р.* Мультиоценочный эвристический алгоритм распределения *M* заданий на *N* процессоров// Моделирование процессов управления. М.:МФТИ, 2004 – С. 170-173.

14. *Гончар Д.Р.* Мультиоценочный эвристический алгоритм распределения M заданий на N одинаковых процессорах // В сб. Процессы и методы обработки информации. М.: МФТИ, 2005. 3 с.

15. *Гончар Д.Р.* Мультиоценочный эвристический алгоритм распределения M заданий на N процессоров. // II Всероссийская научная конференция «Методы и средства обработки информации». М.: МГУ, 2005. С. 537-539.

16. *Гончар Д.Р.* Мультиоценочный эвристический алгоритм распределения M заданий на N процессоров. // Сообщения по прикладной математике. М.: ВЦ РАН, 2005, 16 с.

17. *Гончар Д.Р.* Мультиоценочный алгоритм решения минимаксной задачи составления расписания // Системы управления и информационные технологии № 1.3 (27), 2007. Москва-Воронеж: Научная книга, 2007. С. 324-328.

18. *Фуругян М.Г., Гончар Д.Р.* Мультиоценочный алгоритм решения минимаксной задачи составления расписания // Проблемы управления безопасностью сложных систем: Труды XV-й международной конференции. Часть 2. /ИПУ РАН. – М., 2007 – С. 149-153.

19. *Фуругян М.Г., Гончар Д.Р.* Мультиоценочный алгоритм решения минимаксной задачи составления расписания // Сборник научных трудов– Моделирование процессов обработки информации. М.: МФТИ, 2007. С. 202-209.

ЛИЧНЫЙ ВКЛАД

При построении инструментальной системы «СРВ-Конструктор» были приняты за основу и использованы после соответствующей переработки ряд разработок по предыдущей версии данной системы для ЕС ЭВМ, а именно формальный язык описания пользовательской СРВ для системы «СРВ-КОНСТРУКТОР», система автоматизированного построения компиляторов «Супер», ряд конкретных, специфичных для СРВ технических решений (при организации монитора реального времени, генерации кода и т.п.).

Сама идея создания САПР «СРВ-Конструктор» принадлежит основателю сектора проектирования ВСРВ Сушкову Б.Г. Идеи некоторых эвристических алгоритмов выдвинуты совместно с М.Г. Фуругяном. В совместных работах автору принадлежит теоретическая часть, постановки задач, разработка, испытания и исследования предложенных в диссертации подходов и алгоритмов. О.Л.Кондратьеву, Д.В.Белому, Д.С. Гузу и Д.В. Красовскому принадлежит рассмотрение ряда других алгоритмов управления и анализа систем реального времени, не вошедших в данную диссертацию.

Гончар Дмитрий Русланович

МЕТОДЫ ПЛАНИРОВАНИЯ ВЫЧИСЛЕНИЙ
В САПР СИСТЕМ РЕАЛЬНОГО ВРЕМЕНИ

Подписано в печать 05.06.2008
Формат 60x84 1/16. Бумага офсетная. Усл. печ. л. 1,0.
Тираж 100 экз. Заказ № 17

Вычислительный центр им. А.А.Дородницына РАН
119991, г. Москва, ул. Вавилова, 40