**АННОТАЦИЯ**

В данной выпускной квалификационной работе рассматривается усовершенствованный метод оптимизации решения комплексных задач теории расписаний. Целью данной работы является разработка и исследование иерархической системы и метода построения комплексного расписания обработки партий данных при условии формирования комплектов и задания директивных сроков их выпуска.

Ключевые слова: иерархическая система, конвейерная обработка, оптимизация, требования, партии, расписание, комплекты.

В работе представлена иерархическая модель оптимизации построения расписания обработки партий данных различных типов при условии формирования комплектов и задания директивных сроков их выпуска, методы построения партий данный, метод построения и оптимизации расписания, метод формирования комплектов из результатов обработки данных.

Для доказательства работоспособности этой системы в целом в разделе 4 приведен сравнительный анализ всей системы с работой системы без использования модуля оптимизации составов партий данных, а так же системы с использованием генетического алгоритма для формирования решений по составам партий.

В представленной пояснительной записке к выпускной квалификационной работе 4 основных разделов на страницах, включающих 11 рисунков.

**Оглавление**

[ВВЕДЕНИЕ 5](#_Toc517260087)

[1 ОБОСНОВАНИЕ АКТУАЛЬНОСТИ РАЗРАБОТКИ МЕТОДОВ ОПТИМИЗАЦИИ СОСТАВОВ ПАРТИЙ ДАННЫХ 9](#_Toc517260088)

[1.1 Постановка задачи 9](#_Toc517260089)

[1.2 Анализ современных методов построения комплексных расписаний партий данных 10](#_Toc517260090)

[1.3 Обоснование подхода к решению задачи построения расписаний обработки партий данных разных типов при наличии ограничения на интервал функционирования системы 17](#_Toc517260091)

[Выводы раздела 1 19](#_Toc517260092)

[2 ОБОСНОВАНИЕ МОДЕЛЕЙ ПОСТРОЕНИЯ КОМПЛЕКСНЫХ РАСПИСАНИЙ ОБРАБОТКИ ДАННЫХ В КОНВЕЙЕРНЫХ СИСТЕМАХ 20](#_Toc517260093)

[2.1 Декомпозиция системы обработки партий данных при условии формирования комплектов и задания директивных сроков их выпуска 20](#_Toc517260094)

[2.2 Обоснование модели формирования и обработки партий данных в конвейерных системах 24](#_Toc517260095)

[2.3 Обоснование модели формирования комплектов с учетом требований оперативности и директивных сроков 30](#_Toc517260096)

[2.4 Обоснование оптимизационной модели формирования составов партий данных и расписаний их обработки с учетом формирования комплектов из результатов обработки данных 38](#_Toc517260097)

[Выводы по разделу 2 54](#_Toc517260098)

[3 ОБОСНОВАНИЕ МЕТОДОВ ПОСТРОЕНИЯ КОМПЛЕКСНЫХ РАСПИСАНИЙ В КОНВЕЙЕРНЫХ СИСТЕМАХ 55](#_Toc517260099)

[3.1 Обоснование метода формирования составов партий данных 55](#_Toc517260100)

[3.2 Описание метода формирования составов партий данных 59](#_Toc517260101)

[3.3 Обоснование метода построения расписаний с учетом формирования комплектов 65](#_Toc517260102)

[3.4 Обоснование метода формирования комплектов 68](#_Toc517260103)

[3.5 Описание метода формирования комплектов 71](#_Toc517260104)

[Выводы по разделу 3 73](#_Toc517260105)

[4 АНАЛИЗ ЭФФЕКТИВНОСТИ МЕТОДОВ ОПРЕДЕЛЕНИЯ СОСТАВОВ ПАРТИЙ ДАННЫХ И РАСПИСАНИЙ ИХ ОБРАБОТКИ В КОНВЕЙЕРНЫХ СИСТЕМАХ 74](#_Toc517260106)

[4.1 Обоснование использования генетических алгоритмов при решении задачи формирования составов партий 74](#_Toc517260107)

[4.2 Анализ эффективности метода формирования составов партий с учетом требования оперативности формирования комплектов из результатов обработки 79](#_Toc517260108)

[4.3 Анализ эффективности метода формирования составов партий при условии формирования комплектов в заданные для них директивные сроки 82](#_Toc517260109)

[Выводы раздела 4 84](#_Toc517260110)

[ЗАКЛЮЧЕНИЕ 85](#_Toc517260111)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 87](#_Toc517260112)

[ПРИЛОЖЕНИЕ А 91](#_Toc517260113)

[ПРИЛОЖЕНИЕ Б 112](#_Toc517260114)

# ВВЕДЕНИЕ

**Актуальность темы**. На современном этапе развития теории расписаний и задач оптимизации рассматриваются как приближенные, так и точные методы. В информационных технологиях часто появляется необходимость обработки больших массивов разнотипных данных за ограниченное время. Одним из способов решения данной проблемы является применение систем конвейерного типа. В этих системах важной составляющей является поступление требований на обработку. Теория расписаний является частью исследования операций. Теория расписаний исследует задачи, в которых необходимо упорядочить или, другими словами, определить последовательность выполнения совокупности работ, использования каких-либо средств и т.д.

Задачи упорядочения носят самый общий характер. Они возникают там, где существует возможность выбора той или иной очередности выполнения работ: при распределении работ на производстве, составлении расписания приземления самолетов, составлении расписания движения поездов, обслуживании клиентов в обслуживающих системах и т.д.

Результаты, к которым приводит то или иное упорядочение, существенно отличаются. В ряде практических случаев эти различия принимают стоимостный характер или определяются какой-либо другой величиной.

**Цель и задачи работы**. Целью выпускной квалификационной работы является разработка методов построения решения по оптимизации составов партий данных и расписаний их обработки при условии формирования комплектов и задании директивных сроков их выпуска. Для решения поставленной задачи была введена двухуровневая иерархическая модель. На основании полученной модели были поставлены задачи:

* формирование метода оптимизации составов партий всех типов данных;
* формирование метода оптимизации составов партий фиксированного типа данных;
* формирование метода оптимизации расписаний обработки партий данных с учетом формирования комплектов из результатов обработки.

**Предмет и объект исследования**. В выпускной квалификационной работе рассматривается многоуровневая система теории расписания. Теория расписаний ‑ раздел дискретной математики, занимающийся проблемами упорядочения. В большинстве случаев в теории расписания ставится задача дискретной оптимизации: построить расписание, минимизирующее время выполнения работ, стоимость работ и т.п. Расписание ‑ указание, на каких машинах и в какое время должны обслуживаться требования (выполняться работы).

Выполнение программ в конвейеризированной вычислительной системе требует использования её (системы) ресурсов. В первую очередь такими ресурсами являются процессорное время каждого сегмента конвейера и оперативная память, в которой хранится выполняемая программа и обрабатываемые данные. Чтобы наиболее эффективно с точки зрения использования сегментов конвейера использовать вычислительную систему необходимо планировать запуск программ различных типов, то есть составлять расписания обработки данных разных типов. Так как поступление данных конкретного типа инициирует в конвейеризованной системе выполнение программы фиксированного типа, то под «расписанием» понимается порядок поступления данных разных типов на вход системы или, другими словами, порядок запуска программ разных типов на выполнение. Под комплектом понимаем наборы различных типов обработанных данных, сформированных в единую структуру. Под временем выпуска комплекта понимаем время за которое произойдет обработка всех требований, необходимых для составления комплекта конкретного типа.

**Научна новизна**. В выпускной квалификационной работе рассматривается градиентный подход с использованием жадной стратегии для решения задачи о оптимизации составов партий. Градиентный метод использует понятие окрестности и шага решения. Под окрестностью понимают набор возможных решений для получения локального максимума в текущей окрестности. Под локальным максимумом понимается понятие наилучшего решения среди всех из текущей окрестности. Под шагом алгоритма понимается понятие увеличения окрестности и получения большего числа потенциальных локальных максимумов для нахождения среди них глобального оптимального (наилучшего решения для поставленной задачи).

**Практическое значение работы**. Рассматриваемая система применима в конвейерном производстве. Система позволяет оптимизировать составы партий данных для их обработки для оптимизации формирования комплектов с учетом оперативности и директивных сроков. Система построена таким образом, что может использоваться в любых отраслях, применяющих конвейерную обработку данных. В текущем контексте разработанная программа доказывает необходимость использования полученной системы по сравнению с системой, основанной на фиксированных партиях (составы партий не изменяются) или использующая генетические алгоритмы для получения составов партий данных на верхнем уровне системы. В дальнейшем планируется разработать программный продукт, основанный на полученной системе, конвейерной обработки спутниковых снимков земли (эта задача актуальна, так как обработка большого количества изображений занимает огромное время).

**Структура работы**. Выпускная квалификационная работа состоит из введения, семи основных разделов, заключения, списка использованных источников и четырех приложений.

В первом разделе обоснована актуальность разработки методов оптимизации составов партий данных.

Во втором разделе представлены модель системы и модель оптимизации составов партий данных, модель формирования расписания и модель формирования составов комплектов.

В третьем разделе описаны метод построения и оптимизации порядка обработки партий данных при формировании комплектов с учетом периодичности и директивных сроков.

В четвертом разделе проводится анализ результатов работы разработанных методов оптимизации.

В результате выполнения выпускной квалификационной работы было разработано Windows-приложение, которое строит расписания обработки данных многих типов, с учетом времени перенастройки оборудования с обработки требования одного типа на другой. Разработанная система производит оптимизацию расписания по нескольким критериям эффективности, программный комплекс условно разделён на три уровня. Каждый из них представляет собой поиск оптимума по одному или нескольким критериям.

Из-за того что обработка на всех устройствах происходит последовательно, и обработка каждой партии не может быть разбита на несколько частей, необходимо сформировать порядок поступления партий данных только на первое устройство.

# ОБОСНОВАНИЕ АКТУАЛЬНОСТИ РАЗРАБОТКИ МЕТОДОВ ОПТИМИЗАЦИИ СОСТАВОВ ПАРТИЙ ДАННЫХ

## Постановка задачи

Целью работы является совершенствование методов локальной оптимизации решений по составам партий данных, обрабатываемых в конвейерной системе, и решений по порядкам обработки этих партий на сегментах в конвейерной системе (расписанию обработки партий), а так же решений по формированию комплектов из результатов обработки. Достижение цели обеспечивается декомпозицией обобщенной функции (цели) системы на совокупность иерархически упорядоченных подфункций (подцелей), каждая из которых реализуется на определенном уровне иерархии системы.

В результате выполненной декомпозиции обобщенной функции системы в рассмотрение введена двухуровневая модель иерархической игры определения локально оптимальных решений по составам партий и расписаниям их обработки.

На основании предложенного иерархического подхода достижение сформулированной цели обеспечивается решением следующих задач: а) обоснования математической модели вычислительного процесса обработки партий данных в конвейерной системе; б) обоснования вида критериев оптимизации в модели иерархической игры, используемой для определения эффективных решений по составам партий данных и расписаниям их обработки; в) обоснование метода локальной оптимизации решений по составам партий данных; г) обоснование метода локальной оптимизации решений по расписаниям обработки партий; д) обоснование метода формирования комплектов из результатов обработки партий данных.

## Анализ современных методов построения комплексных расписаний партий данных

Развитие современных методов построения расписаний обработки партий представлено в работах [2-11].

В [2] выполнена классификация задач управления обработкой партий (построения расписаний обработки партий). Задачи различаются по виду процесса обработки (непрерывный либо дискретный), способам представления времени моделирования (непрерывное либо дискретное), способам формирования партий и т.д. Управление обработкой партий предполагает построение расписаний для фиксированного их состава, определение количества и размера партий до реализации процедуры построения расписаний (алгоритм определения составов партий никак не связан с характеристиками оборудования и процессом обработки, составы партий определяются без связи с построением расписаний), определение размеров партий совместно с решением задачи построения расписаний их обработки. Методы, рассмотренные в [2], предполагают формирование партий и распределение их по обрабатывающим приборам при дискретном времени моделирования для непрерывного производства. При этом использован аппарат частично целочисленного линейного программирования (ЧЦЛП). Большая размерность модели (наличие более тридцати ограничений), использование методов ЧЦЛП, задание ограниченного количества обрабатывающих приборов (ограниченного количества ресурсов) делают затруднительным определение решения для задач большой размерности за ограниченное время.

Методы управления обработкой партий, рассматриваемые в [3], предполагают наличие фиксированных партий данных и их обработку на ограниченном количестве приборов (задача определения оптимальных составов партий в этой работе не рассматривается).

В работах [4,5] рассматривается задача управления обработкой партий в непрерывном (химическом) производстве. Под партиями в этом случае подразумевается объемы материалов, участвующих в процессе производства (объемы партий определяются способом производства требуемого вида продукта, количеством конечного продукта, определяемым спросом). Таким образом, в [4,5] реализуется распределение обработки партий материалов различных видов (размер партий не оптимизируется), обеспечивающих выпуск продуктов разных типов, по параллельно действующим машинам. При этом рассматриваются задачи ограниченной размерности (2 продукта, 2-3 прибора в обрабатывающей конвейерной системе).

В работе [6] рассматривается решение задачи среднесрочного планирования выпуска продукции при ограниченном количестве ресурсов и последующего составления расписаний обработки сформированных при планировании партий. При этом под партией подразумевается совокупность изделий одного типа, выпуск которых закрепляется за определенным предприятием или производственным участком. Размеры партий определяются в соответствии с заказами на производство и директивными сроками их выпуска. Модель оптимизации составов партий учитывает только стоимостные параметры выпуска продукции, но не учитывает временные характеристики и особенности технологических процессов. На основе полученного решения по распределению заказов по производственным участкам решается задача выделения для них ресурсов с целью обработки. Совместно задача определения составов партий и управления выпуском (формирования расписаний обработки партий) в работе не решается. В [6] использованы модель большой размерности и аппарат ЧЦЛП, что ограничивает размерность решаемой задачи.

В работе [7] решается задача определения количества и составов партий единичных (разнотипных) требований, обрабатываемых на одном приборе, с заданными директивными сроками обработки и стоимостью доставки партий. Составы партий разнотипных требований формируются с учетом директивных сроков. Задача предполагает наличие одного обрабатывающего прибора, для которого отсутствуют простои при обработке партий. Формирование партий предполагает и одновременное автоматическое построение расписания их обработки. В результате решается задача определения составов партий на основе заданных директивных сроков окончания обработки входящих в них требований.

Аналогичная задача формирования партий требований в соответствии с их директивными сроками завершения, обрабатываемых на параллельных машинах, рассматривается в [8]. Объединение требований в партии реализуется с использованием эвристической процедуры в соответствии со значениями параметров начала обработки и директивными сроками окончания обработки. Т.к. обработка двух партий на одной машине (приборе) не может пересекаться, тогда сроки начала и окончания обработки партий используются для их распределения по параллельным машинам (расписание вытекает из сформированных составов партий).

В работе [9] решается задача планирования производства полупроводников, предполагающая совместное формирование составов партий обрабатываемых пластин разных типов и расписаний обработки партий в конвейерной системе с параллельно действующими машинами. При определении локально оптимальных решений по составам партий и расписаниям их обработки используется один обобщенный критерий. Особенностью аппарата формирования составов партий является использование метода отжига. Реализуется поиск в окрестностях текущего локально эффективного решения. Для формирования новых решений, входящих в окрестности, введены эвристические правила, оперирующие со случайно выбираемыми партиями, изменяющие как составы партий, так и расписания их обработки (изменение позиции партии, изменение обрабатывающего прибора для выполнения операции, создание новых партий). На основе сформированного совместного решения по составам партий и расписаниям их обработки выполняется оценка эффективности расписания с использованием дизъюнктивного графа. Предложенный в [9] подход позволяет реализовывать стохастический поиск локально оптимальных решений.

Планированию производственного процесса посвящена работа [10], в которой решается задача распределения заказов на производство продукции по сменным заданиям, распределения заданий по партиям и формирование расписаний обработки партий. Для определения составов сменных заданий используется эвристическая процедура, для определения составов партий разработана имитационная процедура, позволяющая моделировать прохождение партиями конвейерной системы. Определение эффективных составов партий предполагает задание параметров имитационной модели, соответствующих размерам партий изделий каждого типа, и проведения моделирования. В соответствии с результатами моделирования выбираются значения параметров размеров партий, обеспечивающие минимальное значение критерия. После того как с использованием моделирования определены оптимальные размеры партий, выполняется формирование расписаний их обработки.

В [11] реализуется решение задачи определения составов партий компонент, из которых выполняется формирование элементов (аналог формирования комплектов из обработанных в системе изделий). Для формирования партий и расписаний их обработки введена оптимизационная модель, являющаяся многопараметрической и многоиндексной. Определение решений по количеству и составам партий осуществляется путем полного перебора возможных значений этих параметров. Расписание для полученного решения формируется посредством использования эвристической процедуры. При большой размерности задачи (значительное число типов компонент и количество компонент каждого типа) прямой перебор при формировании партий не обеспечивает решение поставленной задачи за ограниченное время. В итоге решение задачи определения составов партий и построения расписаний их обработки реализуется путем привлечения: 1) аппарата ЧЦЛП (однако при большой размерности задачи получение решения за ограниченное время является затруднительным, решения по составам партий формируются без учета расписаний их обработки); 2) методов формирования партий с учетом директивных сроков окончания обработки входящих в них требований; 3) эвристических процедур и правил (однако, применение правил не позволяет получить решения, приближающиеся к оптимальным).

Т.е. эффективные составы партий определяются с привлечением аппарата ЧЦЛП, что делает затруднительным решение задач большой размерности. Либо составы партий определяются, исходя из директивных сроков окончания обработки данных, что не предполагает оптимизацию. Либо составы партий определяются на основе эвристических правил, что также затрудняет получение эффективных решений. В соответствии с этим разработка моделей и методов определения локально оптимальных составов партий и расписаний их обработки является актуальной задачей.

Целью работы является совершенствование методов локальной оптимизации решений по составам партий данных, обрабатываемых в конвейерной системе, и решений по порядкам обработки этих партий на сегментах в конвейерной системе (расписанию обработки партий). Достижение цели обеспечивается декомпозицией обобщенной функции (цели) системы на совокупность иерархически упорядоченных подфункций (подцелей), каждая из которых реализуется на определенном уровне иерархии системы. В результате выполненной декомпозиции обобщенной функции системы в рассмотрение введена двухуровневая модель иерархической игры определения локально оптимальных решений по составам партий и расписаниям их обработки. На основании предложенного иерархического подхода достижение сформулированной цели обеспечивается решением следующих задач: а) обоснования математической модели вычислительного процесса обработки партий данных в конвейерной системе; б) обоснования вида критериев оптимизации в модели иерархической игры, используемой для определения эффективных решений по составам партий данных и расписаниям их обработки; в) обоснование метода локальной оптимизации решений по составам партий данных; г) обоснование метода локальной оптимизации решений по расписаниям обработки партий.

В рассматриваемой работе реализуется решение задачи обоснования математической модели вычислительного процесса обработки партий данных в конвейерной системе при формировании комплектов из результатов обработки и задании директивных сроков формирования комплектов каждого типа, а также задачи обоснования вида критериев оптимизации в модели иерархической игры, используемой для определения эффективных решений по составам партий данных и расписаниям их обработки.

Задача, близкая к рассматриваемой в данной работе, решалась в [12]. Особенностью постановки задачи, решаемой в [12], является наличие временных ограничений на длительности интервалов времени обработки партий данных, т.е. задание временных интервалов, в течение которых выполняется обработка партий. Из данных, обработанных в течение этих интервалов времени, формируются комплекты установленного состава. Для решения этой задачи в [12] выполнено обоснования метода формирования комплектов из результатов обработки данных, полученных по истечении каждого установленного интервала времени функционирования системы. При этом оценка эффективности решений по составам партий выполняется с точки зрения количества комплектов, сформированных из результатов обработки данных, полученных по истечении заданных временных интервалов.

Декомпозиция обобщенной функции системы управления вычислительным процессом обработки партий при задании ограничений на интервалы времени реализации операций с данными и условии формирования комплектов, выполненная в [12], позволила определить три уровня иерархии принятия решений в системе. На первом уровне осуществляется формирование решений по составам партий данных, на втором – формирование решений по составам групп партии, обрабатываемых в течение временных интервалов заданной длительности, на третьем уровне– формирование решений по расписаниям обработки партий данных в конвейерных системах. Для каждого уровня определены формы критериев эффективности принятия решений и методы формирования соответствующих решений.

Рассматриваемая в предлагаемой работе задача также предполагает определение решений по составам партий данных и порядкам их обработки в конвейерной системе (расписаниям обработки партий в конвейерной системе) при условии формирования комплектов из результатов. Однако ее особенностью является задание директивных сроков формирования комплектов каждого типа. Т.е. постановка задачи предполагает задание количества комплектов каждого типа, которые должны быть сформированы из результатов обработки данных, и задание директивных сроков формирования каждого комплекта определенного типа (моментов времени, к наступлению которых каждый комплект определенного типа должен быть сформирован). В соответствии с этим определение количества сформированных комплектов реализуется не с учетом результатов обработки, полученным к моментам времени окончания заданных интервалов времени (как в работе [12]), а с учетом результатов, формируемых по мере обработки данных (в течение всего времени обработки партий). При этом моменты времени окончания формирования каждого комплекта определенного типа должны соответствовать директивным сроком, для них заданных.

Вследствие рассмотрения выше работ по конвейерной обработки партий данных с учетом формирования комплектов было выяснено что необходимо сформулировать подход к комплексному решению задачи формирования партий данных для обработки в конвейерной системе при формировании комплектов с учетом оперативности и директивных сроков.

## Обоснование подхода к решению задачи построения расписаний обработки партий данных разных типов при наличии ограничения на интервал функционирования системы

Задача построения расписания обработки партий с учетом формирования комплектов при условии оперативности и директивных сроков имеет обобщённую цель – обработать данные в максимально сжатые сроки. В то же время необходимо учитывать наличие директивных сроков на формирование комплектов. Данная задача трудноразрешима. В соответствии с подходом вертикальной декомпозиции описанной в [14] достижение цели может быть представлено в виде иерархии подцелей. Тогда при разбиении обобщённой цели на подцели имеют место следующие свойства иерархической обработки:

* приоритетность решений, это свойство следует из необходимости передачи данных с уровня на уровень;
* зависимость эффективного решения на вышестоящем уровне от решения на нижестоящем уровне;
* достижение обобщённой (внешней, глобальной) цели системы возможно только при достижении всех подцелей (внутренних, локальных целей).

В результате декомпозиции обобщённой цели сформирована (определена) трёхуровневая иерархически упорядоченная структура поиска решения задачи (с локальными подцелями на каждом уровне), изображённая на рисунке 1.1.

Для оценки эффективности решений на каждом уровне должны быть введены критерии оценки, которые должны учитывать:

* на уровне расписания – эффективность использования оборудования конвейерной системы при размещении рассматриваемой партии в последовательностях обработки партий данных;
* на уровне комплектов – формирование порядка комплектов для минимизации времени формирования комплектов партий данных из результатов обработки;
* на уровне партий данных – минимизация времени формирования комплектов из реультатов обработки с учетом оперативности и директивных сроков.



Рисунок 1.1 – Структурная схема системы построения расписаний обработки партий данных

## Выводы раздела 1

Таким образом, решение задачи составления расписаний обработки партий данных разных типов при условии формирования комплектов с учетом оперативности и директивных сроков можно представить в виде двухуровневой иерархической системы с заданием внутренних критериев на каждом из уровней функционирования системы.

# ОБОСНОВАНИЕ МОДЕЛЕЙ ПОСТРОЕНИЯ КОМПЛЕКСНЫХ РАСПИСАНИЙ ОБРАБОТКИ ДАННЫХ В КОНВЕЙЕРНЫХ СИСТЕМАХ

## Декомпозиция системы обработки партий данных при условии формирования комплектов и задания директивных сроков их выпуска

Для построения концептуальной модели системы воспользуемся технологией моделирования потоков данных.

Для начала представим нашу систему в виде одного блока верхнего уровня, решающего глобальную поставленную задачу (рисунок 2.1).

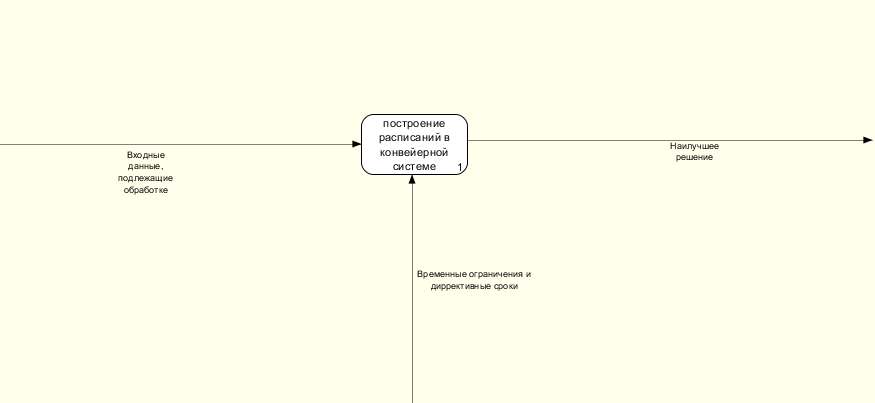


Рисунок 2.1 – Основная цель проекта

Далее произведем декомпозицию основной цели на подцели и покажем их взаимодействие между собой (рисунок 2.2).

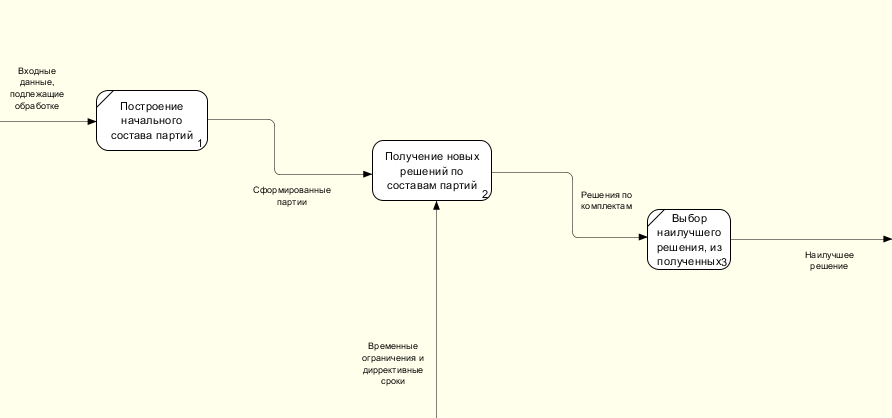


Рисунок 2.2 – Декомпозиция основной целипроекта

Далее произведем декомпозицию центрального узла, отвечающего за формирование новых решений и уточнения наилучшего решения (рисунок 2.3)

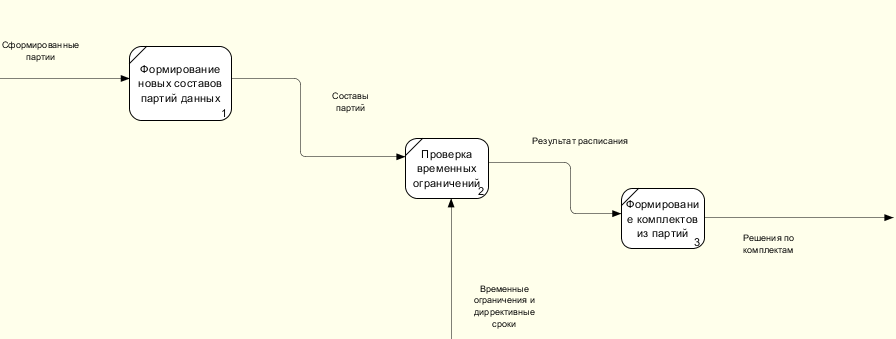


Рисунок 2.3 – Декомпозиция узла оптимизации решений по составам партий

На рисунках продемонстрирована основная концепция системы – построение начального решения и последовательное приближение его к наилучшему решению в заданной окрестности.

На рисунке 2.4 изображена диаграмма Венна, описывающая вложенные уровни проектирования.

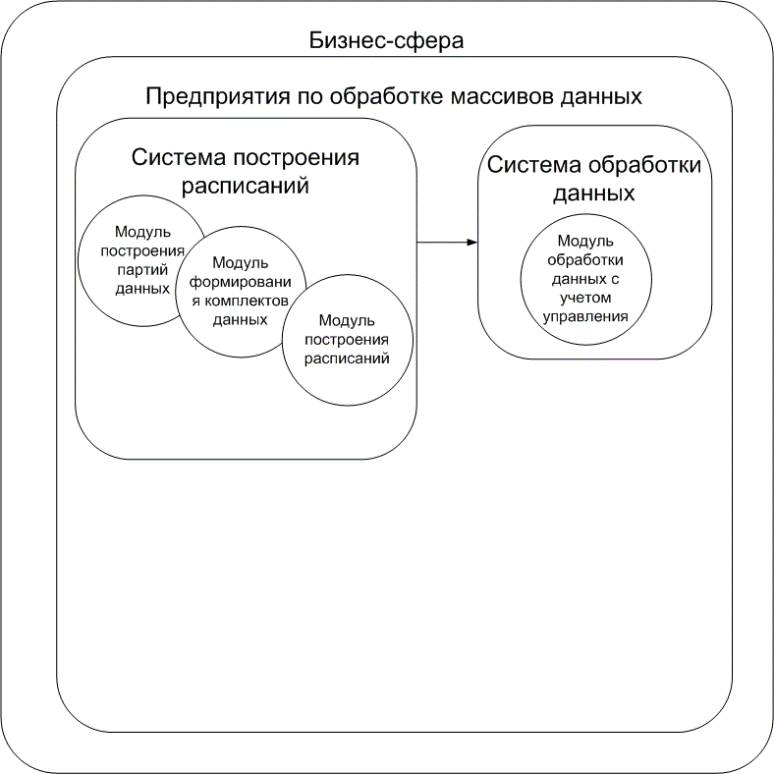


Рисунок 2.4 – Диаграмма Венна, описывающая вложенные уровни проектирования

Вложенными уровнями проектирования рассматриваемой системы является проектирование её отдельных частей и взаимосвязь этих частей.

За систему управления обработкой отвечает система конвейеризации работ обработки данных. Для получения эффективных сроков обработки комплектов данных используются алгоритмы построения расписаний обработки с заданными директивными сроками выпуска комплектов.

Проектируемая система должна включаться в основную систему обработки, получать на вход данные, каталогизировать их и выдавать в виде результата расписание обработки данных для выпуска комплектов за заданное время. Выходные данные должны быть получены за заданное время с заданным отклонением от него.

Модуль управления обработкой напрямую воздействует на систему обработки посредством построенного расписания обработки партий данных, сформированного в модуле построения расписания.

В рамках научно-исследовательской работы рассматривается информационная система по управлению обработкой, в которую включен модуль построения расписаний обработки данных. Целью информационной системы является ускорение обработки комплектов данных на выходе системы.

С учётом особенностей реализации функционирования взаимосвязи подсистем, реализующих технологический процесс системы, представлены на рисунке 2.5.

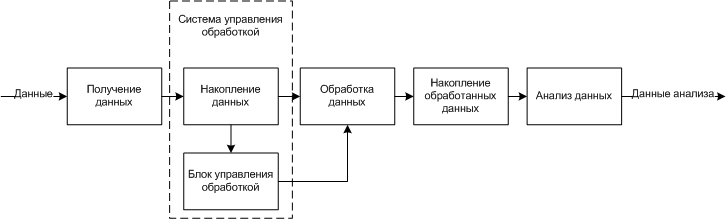


Рисунок 2.5 – Взаимосвязи подсистем с учётом особенностей реализации функционирования

В соответствии с изложенным подходом к декомпозиции обобщенной функции системы на совокупность иерархически упорядоченных подфункций, формирование решений на уровнях системы построения расписаний обработки партий осуществляется следующим образом: первый уровень – решения по количеству и составам партий, а также распределение результатов по комплектам после определения расписаний обработки партий, второй уровень – решения по порядку обработки партий сформированного состава на сегментах конвейера. Выполненная декомпозиция обобщенной функции системы на совокупность иерархически упорядоченных подфункций позволяет рассмотреть задачу как задачу теории иерархических игр.

## Обоснование модели формирования и обработки партий данных в конвейерных системах

Под конвейеризацией выполнения программы подразумевается разделение её на фрагменты, каждый из которых закреплён за соответствующим сегментом конвейера. Маршруты обработки данных всех типов одинаковы, строго фиксированы и предполагают прохождение ими всех сегментов конвейера. Обозначим: *l* – индекс сегмента конвейера (), *n* – количество типов данных, обрабатываемых в системе, *i* – идентификатор типа данных (). Количество элементов в наборе данных, характеризуемых индексом *i*, обозначим как . Данные *i*-того типа () обрабатываются соответствующей программой. В системе используется *n* типов программ, обрабатывающих данные *i-*ыхтипов (). Для формирования решений по составам партий данных введены обозначения: – количество партий данных *i*-го типа (), формируемых на первом уровне принятия решений, элементы образуют вектор *М*, соответствующий количествам партий данных *n* типов; *А*– матрица, элемент которой – это количество данных *i*-го типа в *h*-ой партии (). Решение, формируемое на верхнем уровне иерархии системы имеет вид: [*М*, *А*], где *М*– вектор количества партий данных *i*-ых типов (), *А*–матрица составов партий. В соответствии с решением по составам партий требуется определить последовательности их обработки на сегментах конвейера, т.е. расписание обработки партий. Расписание обработки партий обозначим как , оно представляет собой набор (множество) последовательностей запуска партий на обработку на *l*-ых сегментах конвейера (). Расписание формируется в предположении, что порядок обработки партий является одинаковым на всех сегментах конвейера. Для формализации вида последовательностей расписания в рассмотрении введена матрица порядка обработки партий *P*. Элемент , если партия данных *i*-го типа занимает в последовательности *j*-ю позицию, в случае, если партия данных *i*-го типа не занимает в последовательности *j*-ю позицию, размерность матрицы , где *n* – количество типов данных, – количество партий в последовательностях (). Порядок обработки партий на всех сегментах одинаков, поэтому достаточно определения одной матрицы порядка *P*. В рассмотрение введена матрица *R*– матрица количества данных *i*-ых типов в партиях, занимающих в последовательностях *j*-е позиции (– количество данных *i*-го типа в партии, занимающей *j*-ю позицию в ). Тогда решение, формируемое на нижнем уровне иерархии в системе, имеет вид .



Для определения вида модели вычислительного процесса обработки партий данных введены следующие обозначения:  – время обработки данных i-го типа на *l*-ом сегменте конвейера (); – время переналадки *l*-го сегмента с обработки данных *i*-го типа на обработку данных *k*-го типа; – время первоначальной наладки *l*-го сегмента на обработку данных *i*-го типа;  – матрица длительностей переналадок сегментов конвейера для типов данных, партии которых обрабатываются в системе; – время начала обработки партии данных *i*-го типа, занимающей в последовательности *j*-ю позицию; – матрица моментов времени начала обработки партий данных *i*-ых типов, занимающих в  j-е позиции; – матрица моментов времени начала обработки q-ых данных в партиях, занимающих в  j-е позиции (q – порядковый номер данных в партии в j-ой позиции в (, где – количество данных в партии, занимающей j-ю позицию в ). Элементы матрицы  определяются следующим образом: , где , – момент времени начала обработки первых данных в партии, занимающей j-ю позицию в . Определение значений  и  (,; ; ) выполняется по аналогии с подходом [12].



Для (q=1)-ой позиции данных *i*-го типа в (*j*=1)-ой партии в выражение для определения  формируется при учете параметра  – времени наладки сегмента на обработку данных i-го типа: . Для q>1 (,) имеем [1]: . В этом выражении первое слагаемое определяет длительность наладки сегмента конвейера на обработку данных в первой позиции в , второе слагаемое–длительность обработки данных в первой партии, предшествующих данным в *q*-ой позиции. Для позиции (*q*=1) данных (*j*=2)-ой партии в для  имеем: если – время переналадки первого сегмента конвейера с обработки данных *i*-го типа (первая позиция партии в ) на обработку данных другого типа (вторая позиция партии в ), а – количество данных i-го типа в партии в первой (j=1) позиции в , тогда , где значение  определяется следующим образом:



, где .



Для позиций  () значения  определяются выражением:



, .



Обобщенное выражение для  при  для последовательности  имеет вид: , моменты времени начала обработки q-ых данных в партии с : . Выражения для и  сформированы следующим образом [12]:



;

;



.

Полученные выражения представляют собой модель вычислительного процесса обработки партий данных *i*-ых типов () на *l*-ых сегментах конвейера (). В соответствии с введенными обозначениями вид решения, формируемого на нижнем уровне иерархии системы, примет следующую форму: .



Метод построения расписаний обработки партий данных на втором уровне системы реализует жадный подход к оптимизации. Он предполагает добавление текущей рассматриваемой партии в конец последовательностей (), в которых на предыдущих шагах алгоритма были размещены партии данных, и определение эффективного местоположения этой пар­тии в (определение для рассматриваемой партии позиций в (), которые обеспечивают локально оптимальное решение). Внутренней целью функционирования системы является полное использование временного ресурса сегментов конвейера. Построение решений по порядкам обработки партий на сегментах конвейера в соответствии с этой целью обеспечивает формирование «плотных» расписаний. В этом случае решение по порядку обработки партий должно быть построено таким образом, чтобы временной ресурс системы использовался в полной мере. Тогда критерий эффективности расписаний обработки партий соответствует внутренней цели функционирования системы, определяющей необходимость минимизации простоев оборудования при обработке. При определении текущей эффективной позиции рассматриваемой партии в значение критерия характеризует общие простои всех сегментов конвейера при обработке текущего количества партий, находящихся в  (для всех партий, включенных в последовательности  ()).



В этом случае критерий эффективности расписания обработки партий на нижнем уровне учитывает: а) время простоя сегментов в ожидании начала обработки партий данных (с учетом интервалов наладки, перена­ладки и последующего ожидания); б) время простоя сегментов в ожидании готовности данных при их обработке внутри партий. Простой l-го сегмента в ожидании начала обработки первой в   партии равен значению , суммарное время простоя сегментов в ожидании начала обработки партий в последовательностях  определяется выраже­нием . Интервал простоя l-го сегмента в ожидании начала обработки следующей партии после окончания обработки предыдущей определяется выражением вида: , где *j*>1, – количество данных в предшествующей (j-1)-ой партии. Суммарный простой *l*-го сегмента в ожидании начала обработки всех *j*-ых партий () определён следующем образом: . В этом случае суммарной простой всех *L* сегментов в ожидании начала обработки партий на них (простой между партиями для всех L сегментов) определяется выражением (2.1):



. (2.1)



Простой *l*-го сегмента в ожидании готовности к обработке данных, занимающих *q*-ю позицию в *j*-ой партии в последовательности , определяется выражением вида: , где . Это выражение соответствует интервалу между данными в *q*-ой и (*q*-1)-ой позициях в *j*-ой партии в . Тогда суммарный простой *l*-го прибора в ожидании готовности к обработке всех данных в *j*-ой партии в вычисляется с использованием выражения (2.2):



, (2.2)



где *q* – номер позиции данных в j-ой партии в ,

– число данных в этой *j*-ой партии, .

На основе (4) общий простой *l*-го сегмента в ожидании готовности к обработке данных внутри всех партий в определяется выражением вида: . Суммарный простой всех *L* сегментов конвейера в ожидании готовности данных внутри партий вычисляется выражением (2.3):



. (2.3)

Критерий эффективности решения на нижнем уровне иерархии для текущего количества партий, добавленных в  учитывает общий простой сегментов конвейера при определении порядка обработки этих партий, тогда при его формировании должны быть учтены: выражение  и формулы (2.1) и (2.3).



## Обоснование модели формирования комплектов с учетом требований оперативности и директивных сроков

Особенностью решаемой задачи является необходимость формирования комплектов из результатов обработки данных и задание директивных сроков окончания формирования каждого комплекта определенного типа.

Комплект результатов обработки данных представляет собой совокупность данных, при этом количество данных каждого типа, входящих в комплект одного типа является заданным. Через g обозначим идентификатор типа комплекта, который формируется из результатов обработки данных в конвейерной системе, а через – общее количество типов формируемых комплектов. Через  обозначим количество данных *i*-го типа, которые должны входить в один комплект g-го типа. В рассмотрение введена матрица (W), значения элементов  которой равны числу данных *i*-го типа, которые входят в один комплект g-го типа. Через  обозначим количество комплектов g-го типа, которые должны быть сформированы из обрабатываемых данных i-ых типов () в соответствии с заданием. Тогда количество данных каждого i-го типа (), которые будут обработаны в системе, определяется следующим образом: . В итоге все поступившие на вход системы для обработки данные будут использованы при формировании комплектов разных типов. Также через  обозначим индекс (номер) сформированного комплекта g-го типа, тогда . Т.к. решаемая задача предполагает задание для каждого комплекта директивных сроков его формирования, тогда в рассмотрение введена матрица , элемент которой представляет собой задаваемый момент времени окончания формирования -го комплекта g-го типа (,). При этом для каждого g-го типа элементы  матрицы упорядочены в ее g-ой строке по возрастанию значений моментов времени окончания формирования комплектов: , если . Тогда обработка данных в конвейерной системе направлена на осуществление операций на всех сегментах конвейера с данными разных типов таким образом, чтобы формирование комплектов из результатов обработки выполнялось с учетом (в соответствии) с задаваемыми директивными сроками.



Определение эффективных решений по составам партий данных i-ых типов () вместо фиксированных партий позволяет получить лучший результат с точки зрения формирования в установленные сроки комплектов из результатов обработки. При этом достигается внешняя цель функционирования системы – обеспечение в заданные сроки формирования комплектов установленных типов из результатов обработки данных в конвейерной системе. При этом должно быть определено как количество партий данных, так и их составы.

Входными данными для системы построения расписаний обработки партий при формировании комплектов являются: типы *i* () данных, обрабатываемых в системе, количество данных каждого *i*-го типа; матрица (W), элемент  которой равен количеству данных *i*-го типа, которые должны входить в один формируемый комплект g-ого типа ( ), матрица директивных сроков формирования комплектов, элемент которой соответствует задаваемому моменту времени окончания формирования -го комплекта g-го типа (,). Выходными решениями, обеспечивающими формирование комплектов в установленные сроки, являются: количество и составы партий данных *i*-ых типов (); расписания обработки партий данных в конвейерной системе.



В итоге обобщенная цель функционирования системы представлена как совокупность иерархически упорядоченных целей подсистем (функций, решаемых на иерархически упорядоченных уровнях системы построения расписаний обработки партий). Формирование решений на уровнях системы построения расписаний обработки партий осуществляется следующим образом: первый уровень – решения по количеству и составам партий, второй уровень – решения по порядку обработки партий сформированного состава на сегментах конвейера. Выполненная декомпозиция обобщенной функции системы на совокупность иерархически упорядоченных подфункций позволяет рассмотреть задачу как задачу теории иерархических игр [14,15].

С целью формирования модели иерархической игры для построения комплексных расписаний обработки партий в рассмотрение введены следующие обозначения: – количество партий данных i-го типа (), формируемых на первом уровне принятия решений, при  элементы образуют вектор (М); А– матрица, элемент  которой – это количество данных i-го типа в h-ой партии (), размерность матрицы А – , где . Решение, формируемое на первом уровне системы (количество и составы партии) имеет вид: [М, А].



Расписание обработки партий данных i-ых типов () обозначено как , оно представляет собой совокупность (множество) последовательностей запуска партий на обработку на каждом l-ом сегменте конвейера (). Расписание  имеет вид: . Расписание обработки партий  формируется в предположении, что порядок обработки партий является одинаковым на всех L сегментах конвейера. Для формализации вида последовательностей  расписания в рассмотрении введена матрица  порядка обработки партий в системе; элемент , если партия данных i-го типа занимает в последовательности j-ю позицию,  в случае, если партий данных i-го типа не занимает в последовательности  j-ю позицию, размерность матрицы , где – количество партий в последовательностях  расписания . Порядок обработки партий группы на всех сегментах одинаков, поэтому достаточно определения одной матрицы порядка . В рассмотрение введена матрица – матрица количества данных i-ых типов в партиях, занимающих в последовательностях  j-е позиции (элемент  равен количеству данных i-го типа в партии, занимающей j-ю позицию в , размерность матрицы). Решение, формируемое на нижнем уровне иерархии, имеет вид: .



В соответствии с функциями уровней системы между ними выполняется обмен информацией следующего вида: 1) на вход первого уровня подаются: количество n типов обрабатываемых данных; количество () данных каждого типа, обрабатываемых в системе (определяемое на основе заданных значений  количества комплектов каждого g-го типа ()); матрица (W) составов комплектов; матрица  директивных сроков окончания формирования комплектов g-ых типов (); выхода уровня передаются составы партий данных i-ых типов () – решение [М, А]; 2) на вход второго уровня – решение [М, А]; с выхода уровня – сформированное для решения [М, А] эффективное расписание  обработки партий. При формировании расписаний обработки партий данных i-ых типов состав партий не изменяется (значения  и , поступившие с верхнего уровня, изменены быть не могут). В общем виде модель иерархической игры в соответствии с введенными обозначениями имеет следующий вид [14,15]:



1) первый уровень:

;



2) второй уровень:

.

С точки зрения аппарата теории иерархических игр каждым игроком принимается решение, соответствующее функции (подцели) отдельного уровня. Игрок на верхнем уровне принимает решение по составам партий данных, игрок на втором уровне принимает решение по расписанию обработки партий данных в конвейерной системе.

Комплекты формируются только из результатов обработки данных, которые входят в партии. Полное формирование одного комплекта g-го типа предполагает получение (в требуемом количестве) всех результатов обработки данных, для этого необходимых. Формирование всех результатов обработки данных (в требуемом количестве) требуется реализовать к директивному сроку формирования комплекта соответствующего типа. Поэтому наилучшим решением по составам партий будет являться то решение (с учетом построенного расписания обработки этих партий), которое обеспечивает формирование комплектов разных типов в соответствии с их директивными сроками. Для определения эффективных решений на первом уровне должно определяться соответствие сроков окончания формирования комплектов (для решений по составам партий и расписанию их обработки) заданным директивным срокам. Данное соответствие характеризуется степенью превышения реальных сроков формирования комплектов заданным директивным срокам. Это условие соответствует внешней цели функционирования системы и интерпретируется при формировании критерия эффективности решений на верхнем уровне иерархии.

В соответствии с введенным обозначением вида  определено обозначение для момента времени окончания обработки на l-ом сегменте конвейера q-ых данных в партии, занимающей j-ю позицию в последовательности (), в виде  и аналогичное обозначение для момента времени окончания обработки в системе (соответственно, на L-ом сегменте конвейера) требуемого количества данных i-го типа, включаемых в формируемый -ый комплект g-го типа (,) в следующие виде: . Тогда для -ый комплекта g-го типа (,) может быть определен вектор моментов времени окончания обработки всех данных i-ых типов (в требуемом их количестве при ), обозначенный в виде , формируемый из значений для этих типов. Таким образом, на верхнем уровне с использованием решения по порядкам обработки партий данных вида , сформированного для соответствующего решения по составам партий , определяются вектора (,) значений моментов времени окончания обработки требуемого количества данных i-ых типов . Также для каждого -ый комплекта g-го типа вводится в рассмотрение обозначение  для момента времени окончания его формирования. Значения  определяются следующим образом: , т.е. момент времени окончания формирования -ый комплекта g-го типа определяется как максимум среди моментов времени окончания обработки необходимого количества данных всех i-ых типов (). Тогда в случае  фиксируется запаздывание с формированием -ый комплекта g-го типа относительно его директивного срока . Т.к. цель управления вычислительным процессом состоит в обеспечении формирования комплектов в соответствии с директивными сроками, тогда критерием эффективности решений по составам партий является суммарное запаздывание в формированием всех -ый комплектов g-го типов. Для вычисления суммарного запаздывания с формированием комплектов при обработке данных в рассмотрение введен параметр , соответствующий интервалу времени запаздывания с формированием -го комплекта g-го типа по сравнению с заданным для него директивным сроком. Определение значения  для конкретного комплекта g-го типа выполняется следующим образом (2.4):



(2.4)



Тогда суммарное запаздывание с формированием -ых комплектов g-го типов, соответствующее критерию  принятия решений по составам партий на верхнем уровне, определяется выражением вида:

,

где значение параметра определяется выражением (2.4).



Способ определения значений  моментов времени окончания обработки требуемого количества  данных i-го типа, включаемых в -ые комплекты g-го типов, рассмотрен ниже. Для обоснования способа вычисления значений  моментов времени окончания обработки в системе (на L-ом сегменте конвейера) требуемого количества  данных i-го типа, включаемых в -ые комплекты g-го типов (, ,) , в рассмотрение введены следующие обозначения: 1) – текущий рассматриваемый тип данных, которые из партий распределяются по комплектам в порядке возрастания их идентификаторов g; 2) – счетчик количества данных i’-го типа, включенных в составы всех формируемых комплектов различных типов; 3) – идентификатор комплекта, который формируется на текущей итерации алгоритма; 4) – упорядоченное множество идентификаторов g типов комплектов, формируемое в соответствии с видом матрицы (множество идентификаторов типов комплектов, позиция типа g комплекта в котором определяется в соответствии со значением параметра ); 5) – упорядоченное множество идентификаторов g типов комплектов, являющееся копией множества , используемое при реализации алгоритма; 6) – множество идентификаторов позиций партий данных i-го типа в последовательности , имеющее вид: ; 7) – рассматриваемая позиция партии данных i’-го типа в последовательности ; 8) – порядковый номер комплекта рассматриваемого -го типа.



Таким образом, двухуровневая модель иерархической игры для определения эффективных составов партий данных и расписаний их обработки при формировании комплектов и заданных для них директивных сроках имеет итоговый следующий вид:

1) первый уровень:

, где ,

а значения параметра определяются выражением (2.4);



2) второй уровень:

, где



Дописать то-то

## Обоснование оптимизационной модели формирования составов партий данных и расписаний их обработки с учетом формирования комплектов из результатов обработки данных

С целью решения задачи управления процессом обработки партий данных выполнена декомпозиция обобщенной цели функционирования системы на совокупность иерархически упорядоченных целей подсистем (подфункций, решаемых на иерархически упорядоченных уровнях системы построения комплексных расписаний обработки партий данных). Формирование решений осуществляется на уровнях системы следующим образом: первый– решения по составам партий данных вида [M,A], второй– решения по порядку обработки на сегментах конвейера сформированных партий в виде . При этом составы партий, сформированные на первом уровне, передаются на второй уровень для построения расписаний их обработки, а сформированные эффективные решения по расписаниям обработки партий передаются на первый уровень для формирования из результатов, получаемых при обработке партий, соответствующих комплектов. Т.е. результаты по мере их готовности при завершении обработки каждой из партий распределяются по комплектам. Тогда на первом уровне иерархии подсистем определения решений выполняется решение двух взаимосвязанных задач:



– определения составов партий (решений вида [M,A]);

– формирования комплектов из результатов обработки данных в партиях с учетом определенного на втором уровне иерархии эффективного решения по порядкам их (партий) обработки .

В соответствии с выполненными рассуждениями управление процессом обработки партий данных в конвейерной системе при условии формирования комплектов задаваемых составов предполагает:

1) определение составов партий данных, обрабатываемых в системе (решений вида [M,A]);

2) определение порядка обработки партий данных – расписаний обработки партий в виде ;

3) распределение результатов, полученных при обработке партий данных, по комплектам заданного состава. При этом метод построения расписаний обработки партий отличается от метода, изложенного в разделе 4, т.к. определяемый с его использованием порядок выполнения операций с партиями должен обеспечивать оперативное формирование комплектов. Т.е. метод построения расписаний обработки партий реализует определения такого порядка партий в последовательностях (), который позволяет реализовывать оперативное формирование комплектов.



Рассматриваемая задача является задачей с полной информацией – все параметры, характеризующие обрабатываемые данные (типы данных, ко­личество данных, длительности обработки данных различных типов, составы комплектов различных типов и т.д.) и функционирующую систему (ко­личество сегментов, дисциплина обработки партий, длительности интервалов времени функционирования системы и т.д.) являются заданными.

Для формализации модели процесса обработки партий данных в конвейерной системе при условии формирования комплектов в рассмотрение введены следующие обозначения. Через g обозначен идентификатор типа комплекта, который формируется из результатов обработки данных в конвейерной системе, через – общее количество типов формируемых комплектов. Комплект результатов обработки данных представляет собой совокупность данных, при этом количество данных каждого i-го типа (), входящих в комплект одного (g-го) типа, является заданным. Данные всех n типов входят в определенном количестве в составы комплектов каждого типа. Через  обозначено количество данных i-го типа, которые должны входить в один комплект g-го типа. В рассмотрение введена матрица (W), значения элементов  которой равны числу данных i-го типа, которые включаются в один комплект g-го типа. Т.о. количество результатов обработки данных i-го типа, включаемых в комплект g-го типа, обозначенное через , соответствует i-ой компоненте одного комплекта g-го типа.



Через обозначим количество комплектов g-го типа, которые должны быть сформированы из результатов обрабатки данных i-ых типов () в соответствии с заданием. Т.е количество комплектов заданного типа соответствует количеству однотипных явлений на земной поверхности, которые являются обнаруженными к текущему моменту времени, и для которых должны быть сформированы наборы параметров, характеризующих условия их развития (распространения) одинаковых составов. Тогда количество  данных каждого i-го типа (), которые будут обработаны в системе с целью идентифкации условий развития (распространения) природных явлений и техногенных воздействий, определяется следующим образом: . В итоге все поступившие на вход системы для обработки данные будут использованы при формировании комплектов разных типов. Также через  обозначим индекс (номер) сформированного комплекта g-го типа, тогда .



Входными данными для системы построения расписаний обработки партий при формировании комплектов являются: типы i () данных, обрабатываемых в системе; количество данных  каждого i-го типа, которые должны быть обработаны с целью последующего формирования из результатов комплектов заданных составов; количество типов формируемых комплектов ; количество комплектов каждого типа (); матрица (W), элемент  которой равен количеству данных i-го типа, которые должны входить в один формируемый комплект g-ого типа ( ), матрица () длительностей обработки данных i-ых типов на l-ых сегментах конвейера, матрицы () длительностей переналадок l-ых сегментов конвейера с обработки данных i-го типа на обработку данных j-го типа. Выходными решениями, обеспечивающими формирование комплектов с учетом вводимых в рассмотрение критериев являются: количество и составы партий данных i-ых типов (); расписания обработки партий данных в конвейерной системе.



В соответствии с функциями уровней системы между ними выполняется обмен информацией следующего вида:

1) на вход первого уровня подаются: количество n типов обрабатываемых данных, количество типов комплектов , которые должны быть сформированы из данных, обработанных в партиях, количество комплектов каждого g-го типа  (), количество () данных каждого типа, обрабатываемых в системе (определяемое на основе заданных значений  количества комплектов каждого g-го типа ()); матрица (W) составов комплектов; с выхода уровня передаются составы партий данных i-ых типов () – решение [М, А];



2) на вход второго уровня – решение [М, А]; с выхода уровня – сформированное для решения [М, А] эффективное расписание  обработки партий, на основе которого реализуется распределение результатов между комплектами по мере их (результатов) готовности при окончании обработки каждой из партий. Т.е. в силу требования оперативности идентифкации условий развития (распространения) природных явлений и техногенных воздействий, полученные результаты распределяются по комплектам по мере окончания обработки партий в системе. При формировании расписаний обработки партий данных i-ых типов состав партий не изменяется (значения  и , поступившие с верхнего уровня, изменены быть не могут).



В соответствии с обозначениями  для моментов времени начала обработки q-ых данных в партиях, занимающих в последовательностях обработки  j-ые позиции, определено обозначение  для момента времени окончания обработки на l-ом сегменте q-ых данных в партии, в j-й позиции в последовательности (). По аналогии вводится обозначение для моментов времени окончания обработки в системе (соответственно, на L-ом сегменте конвейера) требуемого количества  данных i-го типа, включаемых в формируемый -ый комплект g-го типа (,) в виде: . Момент времени равен моменту времени окончания обработки партии, результаты из которой позволяют закончить формирование i-й компоненты -го комплекта g-го типа, характеризуемой значением .



Таким образом, – это момент времени окончания обработки в соответствующей партии данных i-го типа, результаты из которой включаются в i-ю компоненту -го комплекта g-го типа таким образом, чтобы их количество в этой компоненте соответствовало значению .



В силу сказанного для -го комплекта g-го типа (, ) определен вектор моментов времени окончания обработки всех данных i-ых типов (в требуемом их количестве  при ), обозначенный в виде , формируемый из значений . Вектор является -ым столбцом матрицы , компонентами которой являются элементы . Т.е. элементами матрицы  являются моменты времени окончания формирования i-ых компонент в количестве  в -ых комплектах g-го типа (размерность матрицы ). Количество матриц  характеризуется значением .



Таким образом, на верхнем уровне с использованием решения по порядкам обработки партий данных вида , сформированного для соответствующего решения по составам партий , определяются матрицы  () значений моментов времени окончания формирования i-ых компонент в -ых комплектах g-ых типов ().



Для каждого -го комплекта g-го типа () вводится в рассмотрение параметр, обозначенный как , соотвествующий моменту времени окончания его формирования. Значения определяются на основе значений  следующим образом: . Т.е. момент времени окончания формирования -го комплекта g-го типа определяется как максимум среди моментов времени окончания формирования необходимого количества данных в каждой его i-ой компоненте ().



Цель управления процессом обработки состоит в обеспечении оперативности получения результатов – условий развития (распространения) каждого природного явления и техногенного воздействия на ОС по мере обработки данных (по мере реализации процесса обработки партий данных), т.е. в оперативности формирования комплектов из результатов обработки партий по мере завершения выполнения в системе операций с ними. Т.о. в случае завершения выполнения обработки партии, получаемые результаты распределяются по комплектам, и для каждого комплекта определяется момент времени его формирования. Т.к. значения для каждого -ый комплекта g-го типа () вычисляются на основе значений элементов  матриц , тогда эти матрицы характеризуют решение по распределению результатов по комплектам, формируемое на основе решения  на втором (нижнем) уровне иерархии.



В общем виде модель иерархической игры для определения составов партий при условии формирования комплектов в соответствии с введенными обозначениями имеет следующий вид:

– первые уровень:

, (2.5)



– второй уровень:

 (2.6)

Т.к. эффективность решений по составам партий определяется с точки зрения оперативности формирования комплектов, а формирование комплектов реализуется на основе решения по порядку обработки партий , тогда модель (2.5), (2.6) может быть представлена в следующем упрощенном виде:

– первые уровень:

, (2.7)



– второй уровень:

. (2.8)

Если для -го комплекта g-го типа данные всех i-ых типов в количестве () подготовлены (т.е. сформированы i-е компоненты комплекта), тогда этот комплект является сформированным в качестве совокупности параметров, характеризующих условия распространения (развития) соответствующего природного явления или техногенного воздействия на ОС. Оперативность формирования каждого -го комплекта g-го типа (,) характеризуется минимальным значением момента времени  (при условии, что результаты из партий распределяются по комплектам по мере окончания их (этих партий) обработки в конвейерной системе).



Выполнение операций со всеми партиями данных i-ых типов () в конвейерной системе начинается в момент времени . Значение  характеризует интервал времени обработки данных в партиях с целью формирования первого комплекта g-го типа (временные затраты на обработку данных в партиях, связанные с формированием первого комплекта g-го типа). Во второй комплект g-го типа могут быть включены данные из таких партий, для которых  (т.е. результаты, полностью формирующие некоторую i-ю компоненту второго комплекта g-го типа, получены при обработке партии в j-ой позиции в (), выполнение операций с которой завершается до окончании формирования первого комплекта g-го типа). Таким образом, результаты обработки данных некоторого i-го типа в партии в j-ой позиции в  () могут быть распределены по i-ым компонентам нескольких комплектов g-ого типа (либо g-ых типов, ). Поэтому интервал времени, затраченный на получение результатов, из которых формируется второй комплект g-го типа, не может быть определен выражением , а характеризуется значением, соответствующим . Аналогичные рассуждения могут быть применены для любого -го комплекта g-го типа. Тогда, временные затраты на обработку данных в партиях с целью формирования всех -ых комплектов () g-го типа определяются выражением . Оперативность формирования комплектов обеспечивается составами партий, для которых значение, получаемое с использованием приведенного выражения является минимальным. Тогда для комплектов каждого g-го типа () требуется обеспечить минимизацию значения времени, затрачиваемого на их формирование. В итоге по всем g-ым типам комплектов необходимо выполнить совместную минимизацию значений временных затрат, связанных с обработкой данных в партиях с целью получения результатов, из которых формируются комплекты. Тогда вид выражения, с использование которого реализуется определение суммарных временных затрат на формирование комплектов g-ых типов () следующий: . Данное выражение представляет собой аддитивную свертку (выражений, определяющих временные затраты на формирование комплектов каждого g-го типа ().



Эффективность применения метода определения составов партий для задачи управления обработкой данных при формировании комплектов определяется путем сравнения получаемых решений с решением для фиксированных партий. В случае обработки фиксированных партий все комплекты формируются одновременно в момент времени окончания обработки последней партии. В этом случае полученное выражение не является приемлемым для определения числовой характеристики решения. Поэтому для идентификации характеристик решений предложен способ, позволяющий определить средние временные затраты на обработку партий данных, связанные с получением результатов, из которых формируются комплекты (средний интервал времени, затрачиваемый на обработку данных в партиях для получения результатов, из которых формируются комплекты). Выражение для определения средних временных затрат на выполнение операций с данными в партиях, из результатов обработки которых формируются комплекты, имеет вид . Предложенное выражение используется в качестве критерия для оптимизации решений по составам партий при реализации требования оперативности и равномерности формирования комплектов из результатов обработки.

Построение расписаний обработки партий на нижнем уровне иерархии реализуется в точки зрения минимизации общего суммарного простоя сегментов конвейера при обработке всех сформированных партий. Для рассматриваемой задачи принятия решений по расписаниям обработки партий предложен метод определения порядка их обработки. Этот метод предусматривает добавление партий в последовательности () таким образом, чтобы обеспечить наиболее быстрое формирование комплектов всех  типов. При этом порядок обработки партий в последовательностях оптимизируется с точки зрения минимизации суммарного времени простоя сегментов конвейера.



Тогда модель двухуровневой иерархической игры, позволяющей реализовать определение составов партий, гарантирующих выполнение требования оперативности формирования комплектов с условиями развития (распространения) природных явлений и техногенных воздействий на ОС, имеет следующий вид:

– первые уровень:

, ; (2.9)

– второй уровень:

, (2.10).



Таким образом, модель (2.9), (2.10) соответствует первой задаче оптимизации решений по составам партий с точки зрения оперативности и равномерности формирования комплектов из результатов обработки.

В отличие от рассмотренной выше задачи вторая задача оптимизации составов партий с учетом формирования комплектов из результатов обработки предполагает задание директивных сроков для каждого -ого комплекта g-го типа (,). Тогда совместное решение сформулированных задач (определения составов партий, расписаний их обработки и формирования комплектов из результатов обработки) должно гарантировать получение комплектов в заданные для них директивные сроки.



Т.к. решаемая задача предполагает задание для каждого комплекта директивных сроков его формирования, тогда в рассмотрение введена матрица , элемент  которой представляет собой задаваемый момент времени окончания формирования -го комплекта g-го типа (,). При этом для каждого g-го типа комплекта элементы матрицы упорядочены в ее g-ой строке по возрастанию значений моментов времени окончания формирования комплектов: , если . Тогда обработка данных в конвейерной системе направлена на осуществление операций на всех сегментах конвейера с партиями данных разных типов таким образом, чтобы формирование комплектов из результатов выполнялось в соответствии с задаваемыми директивными сроками.



Определение эффективных решений по составам партий данных i-ых типов () вместо фиксированных партий позволяет получить лучший результат с точки зрения формирования из результатов обработки комплектов в установленные сроки. При этом достигается внешняя цель функционирования системы – обеспечение в заданные сроки формирования из результатов обработки данных комплектов соответствующих типов.



Входными данными для системы построения расписаний обработки партий при формировании комплектов с учетом задаваемых для них директивных сроков являются: типы i () данных, обрабатываемых в системе; количество типов комплектов , формируемых из результатов обработки; количество комплектов каждого типа (); матрица (W), элемент  которой равен количеству данных i-го типа, которые должны входить в один формируемый комплект g-ого типа ( ); количество данных  каждого i-го типа, которые обрабатываются в системе; матрица директивных сроков формирования комплектов, элемент которой соответствует задаваемому моменту времени окончания формирования -го комплекта g-го типа (,). Выходными решениями, обеспечивающими формирование комплектов в установленные директивные сроки, являются: количество и составы партий данных i-ых типов (); расписания обработки партий данных в конвейерной системе.



В соответствии с изложенным подходом к декомпозиции обобщенной функции системы на совокупность иерархически упорядоченных подфункций, формирование решений на уровнях системы построения расписаний обработки партий осуществляется следующим образом: первый уровень – решения по количеству и составам партий, а также распределение результатов по комплектам после определения расписаний обработки партий, второй уровень – решения по порядку обработки партий сформированного состава на сегментах конвейера. Выполненная декомпозиция обобщенной функции системы на совокупность иерархически упорядоченных подфункций позволяет рассмотреть задачу как задачу теории иерархических игр.

В соответствии с функциями уровней системы между ними выполняется обмен информацией следующего вида: 1) на вход первого уровня подаются: количество n типов обрабатываемых данных; количество  () данных каждого типа, обрабатываемых в системе (определяемое на основе заданных значений  количества комплектов каждого g-го типа ()); матрица (W) составов комплектов; матрица  директивных сроков окончания формирования -ых комплектов g-ых типов (); параметры, определяющие временные характеристики длительностей обработки данных и длительностей переналадки сегментов конвейера на обработку данных каждого типа; с выхода уровня передаются составы партий данных i-ых типов () – решение [М, А]; 2) на вход второго уровня – решение [М, А]; с выхода уровня – сформированное для решения [М, А] эффективное расписание  обработки партий, на основе которого реализуется распределение результатов обработки данных по комплектам заданных составов. В общем виде в соответствии с введенными обозначениями модель иерархической игры имеет форму, аналогичную (2.7), (2.8).



Комплекты формируются только из результатов обработки данных, которые входят в партии. Полное формирование одного комплекта g-го типа предполагает получение (в требуемом количестве) всех результатов обработки данных, для этого необходимых. Формирование результатов обработки данных (в требуемом количестве), являющихся компонентами комплекта, требуется реализовать к директивному сроку окончания его формирования. Поэтому наилучшим решением по составам партий будет являться решение (с учетом построенного расписания обработки партий), которое обеспечивает формирование комплектов разных типов в соответствии с их директивными сроками. При определении эффективных решений на первом уровне должно идентифицироваться соответствие сроков окончания формирования комплектов (для решений по составам партий и расписаниям их обработки) заданным директивным срокам. Данное соответствие характеризуется степенью превышения реальных сроков формирования комплектов заданным директивным срокам. Это условие соответствует внешней цели функционирования системы и интерпретируется при формировании критерия эффективности решений на верхнем уровне иерархии.

С использованием значений элементов матриц  (), каждое из которых представляет собой момент времени окончания формирования i-ой компоненты -го комплекта g-го типа, идентифицируются моменты времени формирования каждого -го комплекта g-го типа, обозначенные  (,). Значения определяются следующим образом: , т.е. момент времени окончания формирования -ый комплекта g-го типа определяется как максимум среди моментов времени окончания обработки необходимого количества данных всех i-ых типов (), т.е. в каждой i-ой компоненте. Тогда в случае фиксируется запаздывание с формированием -го комплекта g-го типа относительно его директивного срока . Для некоторого -го комплекта g-го типа запаздывание с его формированием определяется выражением вида: , которое обеспечивает вычисление требуемого значения в случае . Для определения вида критерия в рассмотрение введен параметр , соответствующий интервалу времени запаздывания с формированием -го комплекта g-го типа по сравнению с заданным для него директивным сроком . Т.е. значение  определяется выражением вида: . Тогда для всех комплектов g-го типа общее запаздывание с их формированием будет определено выражением . В итоге суммарное запаздывание с формированием всех -ых комплектов g-ых типов (,) определяется выражением вида: , которое используется в качестве критерия на верхнем уровне принятия решений по составам партий.



В силу выше сказанного, на нижнем уровне принятия решений по расписаниям обработки партий реализуется метод определения порядка обработки, который предусматривает добавление партий в последовательности  () таким образом, чтобы обеспечить наиболее быстрое формирование комплектов всех типов, и определение порядка выполнения операций с ними в последовательностях  с точки зрения минимизации суммарного времени простоя сегментов конвейера.



Тогда, двухуровневая модель иерархической игры для определения эффективных составов партий данных и расписаний их обработки при формировании комплектов и заданных для них директивных сроках имеет итоговый следующий вид:

– первый уровень:

, ; (2.11)



– второй уровень:

..,. (2.12)

Использование моделей (2.9), (2.10) и (2.11), (2.12) позволяет выполнить оптимизацию составов партий и расписаний их обработки с учетом требования оперативности формирования комплектов из результатов и соответствия моментов времени формирования комплектов заданным для них директивным срокам.

## Выводы по разделу 2

В данном разделе была рассмотрена модель двухуровневой системы конвейерной обработки партий при условии формирования комплектов. На основе модели были предложены и разработаны модель формирования составов партий всех типов данных и метод оптимизации составов партий; модель формирования расписаний при условии формирования комплектов; модель формирования составов комплектов с учетом требования оперативности и директивных сроков. На основе этих методов в разделе 3 описаны соответствующие методы и представлены их описания.

# ОБОСНОВАНИЕ МЕТОДОВ ПОСТРОЕНИЯ КОМПЛЕКСНЫХ РАСПИСАНИЙ В КОНВЕЙЕРНЫХ СИСТЕМАХ

## Обоснование метода формирования составов партий данных

Для обоснования метода формирования локально оптимальных решений по составам партий данных *i-*ых типов () в рассмотрение введены следующие обозначения: – идентификатор типа данных, составы партий которого изменяются на текущем шаге алгоритма; – количество партий данных -го типа, составы которых определяются; *s*– индекс текущего улучшаемого решения путем поиска более эффективного решения в рамках окрестности; *g –* индекс шага алгоритма, выполняемого по отношению к шагу *s*, который соответствует новому формируемому решению, находящемуся в окрестности (*(s+g)*– номер шага алгоритма по формированию решения в окрестности  текущего локально оптимального решения); – индекс партии, состав которой изменяется при реализации алгоритма на текущей итерации; *I –* множество типов данных, для которых выполняется формирование составов партий (первоначально ); – множествотипов данных, используемое при формировании локально оптимального решения по составам партий; *A’*– матрица (аналог матрицы составов партий *А*), используемая при определении наилучшего решения в окрестности  текущего локально оптимального решения (матрица *А’* используется при формировании промежуточных решений); и  ()– матрицы, предназначенные для хранения (буферизации) составов партий данных *i*-ых типов, сформированных на различных итерациях алгоритма (в матрицах  () хранятся решения по составам партий данных *i-*ых типов, входящие в окрестность  с меньшей метрикой, в матрицах  () хранятся решения по составам партий данных *i-*ых типов, входящие в окрестность , сформированные на основе решений из окрестности , т.е. на основе решений из матриц ()); – количество решений по составам партий данных *i*-го типа в матрице  размерностью ; – количество решений по составам партий данных *i*-го типа, в матрице  размерностью ; – индекс текущей строки (решения по составам партий *i-*го типа) в матрице  (); – индекс текущей строки (решения по составам партий *i*-го типа) в матрице  (); – параметр, предназначенный для хранения номера строки в– номера решения, гарантирующего максимальное по модулю значение левого дискретного градиента критерия  [13]; *G–* максимальное по модулю значение левого дискретного градиента , достигаемое в окрестностях различного вида текущего локально оптимального решения (максимального по модулю значение ).



Для текущего локально оптимального решения по составам партий данных *i-*ых типов рассматриваются два вида окрестностей. Построение решений, включаемых в окрестность первого вида, связано с: а) изменением составов партий данных каждого *i-*го типа в заданном количестве  () при неизменных составах партий данных других типов; б) увеличением количества  партий данных *i-*го типа в случае, если построение решений путем изменения составов партий данных этого типа в количестве  является невозможным (выполнены условия окончания формирования составов партий в количестве). Тогда реализуется изменение количества партий данных *i-*го типа и задание начального решения для этого количества партий. Построение решений, включаемых в окрестность второго вида, предполагает совместное использование сформированных решений по составам партий данных всех типов при заданном количестве партий  (реализуется совместное рассмотрение решений по составам партий данных разных типов). Для обоснования алгоритма определения эффективных решений сформулирован способ изменения составов партий данных *i’-*го типа.



Для обоснования способа формирования решений по составам партий данных введены следующие условия и рассуждения (для *i-*го типа данных):

1) количество данных -готипа в партиях должно быть не менее 2 (,); если при формировании начального решения по составам  партий данных *i*-готипа для *h*=1 (первая партия) получено , тогда исследование решений по составам партий данных этого *-*го типа не выполняется;



2) способ формирования начального решения для количества  партий данных *i-*го типа предполагает, что (), а элемент  определяется как ; в дальнейшем при формировании составов партий данные извлекаются из партии с *h*=1 и распределяются по остальным партиям ();



3) значения параметров , задаваемые первоначально для данных всех *i-*ых типов (), равны 2 ();



4) модификация количества партий -го типа данных предполагает, что параметр  увеличивается (при определении составов партий) до тех пор, пока в начальном решении для  выполняется условие ; при условии  (для начального решения при значении ) формирование составов партий -го типа прекращается;



5) формирование решений по составам партий предполагает увеличение количества данных в партии с индексом  (при неизменном составе *h*-ых партий ) и уменьшение количества данных в партии с *h=1* (); при выполнении условия   модификация составов партий продолжается; при выполнении  полученное решение по составам партий в количестве не рассматривается, т.к. является аналогичным полученному на предыдущих шагах алгоритма.



Способ модификации составов партий *i’-*го типа предусматривает, что на основе решений, находящихся в окрестности  первого вида текущего локально оптимального, формируются решения, находящиеся в окрестность  с большей метрикой. Метрика *k* окрестности , содержащей решения по составам партий *i’*-го типа, определяется выражением , где – элемент *i’*-ой строки матрицы , соответствующей текущему локально оптимальному решению по составам партий данных, – элемент *i’*-ой строки матрицы , соответствующей решению по составам партий в окрестности .



В матрице выполняется хранение решений, находящихся в окрестности первого вида , на основе которых реализуется формирование решений, входящих в окрестность  (в матрице хранятся решения, используемые при формировании решений в окрестности). В матрице  реализуется хранение сформированных решений, находящихся в окрестности . Тогда на основе одного решения по составам партий данных (-ой строки матрицы ), формируется совокупность решений, хранимых в матрице , находящихся в окрестности . Если для текущего рассматриваемого решения (-ой строки матрицы ) выполняется условие , тогда увеличение на 1 значения  (при неизменном значении ), а затем увеличение на 1 значения (при неизменном значении ) обуславливает получение одинаковых решений по составам партий данных *i-*го типа. Таким образом, формулировка условия, ограничивающего количество решений в окрестности , имеет следующий вид: если при формировании нового решения по составам  партий данных -го типа на основе решения, представленного строкой  матрицы , для -ой строки выполняется условие , тогда не требуется выполнять увеличение значения элемента ; должен быть выполнен переход к элементу , который увеличивается в случае выполнения условия .



## Описание метода формирования составов партий данных

Алгоритм определения локально оптимального решения по составам партий данных всех *i-*ых типов () содержит следующий порядок шагов:

1) задание для каждого *i-*го типа данных значений параметров  () равными 1 (– количество решений по составам партий данных *i*-ых типов, сформированных на текущей итерации, хранимых в матрице  (буферизируемых в матрице ));



2) формирование начального решения по составам партий данных *i-*ых типов (): а)  (); б) инициализация *h*-ых элементов *i*-ых строк матрицы *А*: (), ; в) проверка условия корректности решений по составам партий данных -ых типов (): и  ; г) если для какого-либо *i*-го типа данных () в начальном решении это условие не выполняется, тогда для данных этого типа формируется фиксированная партия: , , ; модификация множества *I* типов данных: ; параметр *G* инициализируется значением 0; инициализация множества :; результатом является сформированное начальное решение по составам партий данных ;



3) передача решения на второй уровень для формирования расписания обработки партий; получение со второго уровня расписания для текущего решения по составам партий  в виде ; определение значения  для решения ;



4) определение -го типа данных, составы партий которого изменяются: ; ;



5) для определения модифицированных составов партий данных *i’-*го типа (формирования решений в окрестности  текущего локально оптимального решения ) реализуется инициализация: параметра , элементов матрицы  для *i’-*го типа данных ();



6) определение решений по составам партий данных *i’*-го типа, входящих в окрестность  первого вида, формируемых на основе решений, находящихся в матрице (для этого вызывается алгоритм формирования составов партий данных *i’*-го типа, рассмотренный выше, использующий решения, хранящиеся в матрице , для формирования новых решений в окрестности  первого вида); при выполнении для рассматриваемого *i’-*го типа данных условия  реализуется переход на шаг 14;



7) если для данных *i’*-го типа после формирования решений в окрестности первого вида выполняется условие  (в окрестности  текущего локально оптимального решения не получены решения, для которых выполняется условие корректных составов партий), тогда реализуется модификация количества партий: ; выполняется формирование начального решения по составам партий в количестве : осуществляется инициализация значений ; , , , элементов матрицы :  (), ;



8) проверка корректности сформированного начального решения по составам партий данных в количестве : ; (); при его выполнении реализуется переход на шаг 9; если условие корректных составов  партий для начального решения не выполняется, тогда формирование новых решений по составам партий данных *i’*-го типа является невозможным, текущая *i’*-ая строка матрицы *A* соответствует эффективному решению по составам партий данных этого типа; удаление *i’*-го типа данных из множества *I*:  (формирование решений по составам партий данных для *i’-*го типа данных прекращается), переход на шаг 4;



9) анализ эффективности решения по составам партий данных в количестве (решение в окрестности  первого вида, ), для этого выполняется инициализация *i*’-ой строки матрицы *А’* решением по составам партий данных, хранимым в строке матрицы : ( при ); передача сформированного решения  для определения расписания обработки;



10) получение для рассматриваемого решения  со второго уровня расписания обработки партий в виде , его использование для вычисления значения критерия ;



11) для решения *[M(s+g),A’(s+g)]* вычисление левого  (либо правого ) дискретного градиентов целевой функции  [14];



12) при условиию  сформированное решение *[M(s+g),* *A’(s+g)]* не является более эффективным, чем решение (для количества партий ), тогда реализуется переход на шаг 14 (параметр  не инициализируется);



13) если для сформированного начального решения по составам партий (строки  матрицы ) выполняется условие , тогда реализуется сравнение значений и *G*– градиента целевой функции ; если , тогда текущее решение по составам партий данных *i’*-го типа является наилучшим среди рассмотренных решений по составам партий данных в окрестности первого вида; если, тогда значения всех параметров  () обнуляются, индекс  решения по составам партий данных *i’*-го типа (-ой строки  матрицы ) сохраняется: ; изменение значения *G*: , переход на шаг 14;



14) осуществляется проверка условия ; в случае его выполнения реализуется переход на шаг 4; при  (для каждого *i*-го типа данных () сформированы новые решения по составам партий) осуществляется переход на шаг 15;



15) проверка условия , при его выполнении отсутствуют *i-*е типы данных (), для которых сформированные решения удовлетворяют условию корректного состава партий (отсутствует возможность анализа новых решений), выполняется переход на шаг 24; при  для *i-*ых типов данных () реализуется проверка условия :; при выполнении для *i’-*го типа данных () условия решение по составам его партий (при неизменных составах партий других *i*-ых типов (, )) обеспечивает максимальное уменьшение значения целевой функции в рамках окрестности  первого вида; реализуется переход на шаг 16; в случае, если для всех *i-*ых типов данных () выполняется условие , тогда в рамках окрестности первого вида не определено решение, улучшающее решение ; переход на шаг 17;



16) формирование локально оптимального решения  с учетом составов партий данных *i’-*го типа, для которого : инициализация *i’*-ой строки матрицы *A* текущего решения  значениями элементов -ой строки матрицы : (); в результате получено новое локально оптимальное решение ; инициализация  ();; переход на шаг 4;



17) формирование решений по составам партий данных *i*-ых типов, включаемых в окрестность  второго вида, путем комбинирования решений по составам партий данных *i-*ых типов, находящихся в матрицах ; для этого вызывается алгоритм формирования составов партий данных *i*-ых типов, рассмотренный выше, использующий решения, хранящиеся в матрицах , для формирования новых решений в окрестности  второго вида и определяющий лучшее из них в этой окрестности;



18) для *i-*ых типов данных () проверка условий ; при выполнении этих условий в окрестности  второго вида не сформировано решение лучшее, чем текущее локально оптимальное решение ; тогда реализуется переход на шаг 20; если для *i-*ых типов () условия не выполняются, тогда в окрестности  второго вида сформировано решение, являющееся лучшим, чем текущее локально оптимальное решение ; реализуется переход на шаг 19;



19) выполняется формирование нового локально оптимального решения с учетом составов партий данных *i*-ых типов, для которых ; для каждого *i* идентификатора типа данных, для которого выполняется условие  реализуется: инициализация *i*-ой строки матрицы *A* решения значениями элементов -ой строки  матрицы : при ; результат– новое локально оптимальное решение ; инициализация (), ; переход на шаг 4;



20) инициализация множества типов данных :;



21) идентификация *i’-*го типа данных, решения по составам партий данных которого буферизируются в матрице : , ;



22) инициализация строк матрицы  решениями по составам партий данных *i’-*го типа, находящимися в матрице , в количестве :  (, , ,), присваивание ; в результате сформирована матрица , содержащая решения, на основе которых будут формироваться решения в окрестности первого вида с большей метрикой;



23) проверка условия , если условие выполняется, тогда реализуется переход на шаг 21; в случае выполнения условия  промежуточные решения по составам партий данных всех *i-*ых типов () буферизированы в соответствующих матрицах ; осуществляется присваивание , *k=k+1* , реализуется переход на шаг 4;



24) окончание алгоритма.

## Обоснование метода построения расписаний с учетом формирования комплектов

В силу используемого предположения о том, что партии в сформированном расписании имеют одинаковый порядок обработки во всех последовательностях () для отдельной рассматриваемой группы  и того, что количество типов данных ограничено, тогда является ограниченным и количество возможных решений. Для формирования расписаний обработки партий данных в системе использован метод окрестности (метод поиска лучшего решения в рамках некоторых окрестностей текущего локального оптимального решения). Для обоснования алгоритма определения эффективного порядка обработки партий данных в последовательностях  (), образующих расписание  отдельной группы , в рассмотрение введены следующие обозначения: s-индекс (номер) шага алгоритма, на котором зафиксировано текущее локально оптимальное решение по порядку обработки партий в группе ();  – количество партий данных, размещенных в последовательностях  () на предшествующих и текущем s-ом шагах алгоритма;  – вектор, *i*-ый элемент которого – это количество партий данных *i*-го типа, размещенных в последовательностях () на предыдущих и текущем s-ом шагах алгоритма; g– количество выполненных шагов по изменению локально оптимального решения  для получения решения , соответствующего новому порядку партий в    
() (т.е. индекс шага алгоритма по определению лучшего решения в окрестности текущего локально оптимального); – метрика окрестности локально оптимального решения , используемая для идентификации отличия сформированного решения  от решения ; – максимально возможная метрика окрестности исходного решения , в которой выполняется поиск более эффективного решения;  – окрестность решения , в которой выполняется поиск более эффективного решения;  – индекс элемента вектора , соответствующего составам партий данных *i*-го типа в группе , который содержит количество данных в партии, добавляемой в последовательности (), *v* – индекс текущего рассматриваемого столбца матриц  и , в *i*-ой строке которого значения элементов  и соответствуют характеристикам рассматриваемой партии данных *i*-го типа, эффективное местоположение которой в последовательностях  определяется на текущей итерации алгоритма (, ); – индекс столбца матриц  и , в котором первоначально размещаются характеристики рассматриваемой партии данных *i*-го типа при добавлении этой партии в последовательности (, ).



Вектор используется при определении количества  партий данных различных типов, размещенных в последовательностях (). Первоначальная инициализация элементов () выполнена следующим образом:  (). Значение количества партий , размещенных в  (), определяется выражением: . Расчет метрики  окрестности решения : , где – количество типов данных, партии которых включены в рассматриваемую группу , – общее количество партий данных разных типов, размещенных в последовательностях  (). Метрика определяет число элементов в матрице  (число позиций в последовательностях ), в которых изменены значения по отношения к виду матрицы , соответствующей локально оптимальному решению . Начальное решение  по определению позиции партии данных *i*-го типа в последовательностях () формируется путём добавления -ой партии данных *i*-го типа в количестве ) в конец этих последовательностей. Через *g* обозначен номер промежуточного решения , которое формируется на основе исходного , находящегося в окрестности , характеризуемой метрикой . Формирование решения предполагает изменение положения рассматриваемой партии в последовательностях  относительно ее положения, которое соответствует решению .



## Обоснование метода формирования комплектов

Распределение результатов обработки данных по комплектам реализуется с учетом порядка обработки партий в расписаниях, сформированных для определенного решения по их составам. Т.е. комплекты формируются по мере готовности результатов, полученных при обработке партий данных. При реализации распределения результатов обработки по комплектам определяются значения  (,) и значения соответствующих критериев, характеризующих решения по составам партий данных.

На нижнем уровне принятия решений реализуется метод определения порядка обработки партий, предусматривающий добавление партий в последовательности () таким образом, чтобы обеспечить наиболее быстрое формирование комплектов всех  типов. Поэтому для сформированного решения по порядкам обработки партий реализуется распределение результатов по комплектам по мере окончания выполнения операций с ними в конвейерной системе. С целью обеспеения оперативности формирования комплектов из результатов обработки и удовлетворения требования формирования комплектов в соответствии с их директивными сроками введены способы предварительного упорядочивания идентификаторов комплектов. Т.е. в зависимости от вида решаемой задачи (определения составов партий при условии оперативного формирования комплектов, определения составов партий при условии формирования комплектов с учетом заданных для них директивных сроков) идентификаторы типов комплектов g () и номера  () комплектов соответствующих типов будут упорядочиваться различным образом.



Для первой задачи введены в рассмотрение следующие обозначения, что позволит сформулировать метод упорядочивания идентификаторов комплектов с целью их последующего формирования из результатов обработки: – исходное множество идентификаторов типов комплектов, которые формируются из результатов обработки данных (первоначальлный вид множества ); – множество идентификаторов типов комплектов, упорядоченное в соответствии с рассмотренным ниже способом; – идентификатор типа комплекта, выделяемый из множества  в соответствии с введенным в рассмотрение признаком и добавляемый в множество ; *h*- позиция идентификатора типа комплекта, на которой он размещается в множестве ;  – идентификатор типа комплекта, занимающий в множестве  h-ю позицию; – параметр, обратный относительной сложности формирования комплекта g-го типа, вычисляемый по формуле  (т.е. большее значсение параметра  соответствует простому для формирования типу комплекта, меньшее значение соответствует сложному для формирования типу комплекта). Введимый в рассмотрение способ реализует формирование упорядоченного множества  типов комплектов на основе исходного множества .



Первый этап способ формирования упорядоченного множества типов комплектов имеет следующую последовательность шагов:



1) формирование множества  в исходном виде: ;

2) задание исходного значения позиции h в множестве  для рассматриваемого типа данных равной  ();



3) среди типов данных выбирается такой тип данных , для которого выполняется следующее условие: ;



4) тип комплектов добавляется в множество  на позицию h и исключается из множества : , ;



5) модификация значения h позиции типа данных  в множестве : h=h-1; если выполняется условие , тогда реализуется переход на шаг 3;



6) в случае  выполнена первоначальное упорядочивание типов комплектов g в множестве .

В результате реализации рассмотренной последовательности шагов типы комплектов g упорядочиваются в множестве в соответствии с значениями , т.е. множество  определено в виде , где типы комплектов и  размещаются на h-ой и (h+1)-ой позициях в этом множестве в том случае, если . После формирования с использованием рассмотренного способа первоначального вида множества , реализуется уточнение h-ых позиций типов комплектов  при выполнении условия для подряд удущих идентификаторов  и  вида: . Уточнение (изменение) позиций h и (h+1) для типов комплектов  и , для которых выполняется в соответствии со следующей последовательностью шагов:



1) в множестве  определяются такие типы комплектов  и , для которых выполняется условие ;



2) для определенных таким образом типов комплектов  и  выполняется вычисление значений параметров и ; в том случае, если для параметров  и выполняется условие , тогда для рассматриваемых типов комплектов  и производится обмен их позициями в множестве .



3) приведенные действия выполняеются для всех типов комплектов, для которых выполняется условие .

В итоге на основе множества  вида формируется новое множество  следующего вида .



## Описание метода формирования комплектов

Алгоритм определения моментов времени окончания обработки данных i-ых типов в соответствии с расписанием вида , используемых при формирования -ых комплектов g-ых типов, имеет следующий порядок шагов:



1) для g-ых типов комплектов () задание значения параметра  равным 1 ();



2) задание индекса i’ типа данных, которые будут распределяться по комплектам g-ых типов (): , где I – множество типов данных, партии которых обрабатываются в системе; ;



3) задание значения счетчика  количества данных i’-го типа, добавленных в -ый комплект g’-го типа, равным 0 ();



4) задание идентификатора g’ вида комплектов, в которые будут включаться данные i’-го типа: ; ;



5) определение позиции j’ в последовательности  партии i’-го типа, данные из которой будут включаться в -ый комплект g’-го типа: , где h – порядковый номер позиции j партии i’-го типа в множестве ; ; инициализация значения индекса (номера) данных q в партии в j’-ой позиции в : ;



6) если , тогда выполняется переход на шаг 7; если , тогда выполняется переход на шаг 8;

7) если , тогда ; ; выполняется переход на шаг 6; если , тогда переход на шаг 5;



8) определение значения: ; изменение значения номера  формируемого комплекта g’-го типа: ;



9) проверка условия ; в случае его выполнения – переход на шаг 10; при выполнении условия  задание идентификатора вида комплекта g’, в который будут включаться данные i’-го типа: ; ; задание значения счетчика  количества данных i’-го типа, добавленных в -ый комплект g-го типа равным 0 (); реализуется переход на шаг 6;



10) проверка условия , при его выполнении реализуется переход на шаг 11; при выполнении условия  осуществляется модификация идентификатора типа i’, данные которого распределяются по комплектам g-ых типов ():, ; реализуется переход на шаг 3;



11) остановка алгоритма;

## Выводы по разделу 3

В данном разделе были рассмотрены методы конвейерной обработки партий при условии формирования комплектов. Было проведено обоснование метода формирования составов партий данных, метода формирования расписаний с учетом формирования комплектов и метода формирования комплектов с учетом оперативности и директивных сроков. Так же представлены описания обоснованных методов. На основе этих методов была написана программа и в разделе 4 описаны результаты эффективности использования представленных методов в конвейерной системе обработки партий данных.

# АНАЛИЗ ЭФФЕКТИВНОСТИ МЕТОДОВ ОПРЕДЕЛЕНИЯ СОСТАВОВ ПАРТИЙ ДАННЫХ И РАСПИСАНИЙ ИХ ОБРАБОТКИ В КОНВЕЙЕРНЫХ СИСТЕМАХ

## Обоснование использования генетических алгоритмов при решении задачи формирования составов партий

Для обоснования способа формирования решений по составам партий данных введены следующие условия и рассуждения (для некоторого *i-*го типа данных):

1) количество данных -готипа в партиях не может быть менее 2 (,);



2) минимальное количество  партий, которые могут быть сформированы из данных в количестве , составляет две единицы; при этом в одну из партий включаются 2 единиц данных (т.е. ), а состав второй партии предполагает включение в нее оставшихся данных (т.е. );



3) максимальное количество  партий, которые могут быть сформированы из данных в количестве определяется следующим образом: , где через  обозначено округление в меньшую сторону.



Тогда количество данных *i-*го типа в каждой из партий может изменяться от 2 до . В случае максимального количества  партий данных *i-*го типа в каждой из них может быть по 2 элемента этих данных. С другой стороны минимальное количество партий данных равно двум. При этом одна из партий содержит 2 элемента данных, другая  соответственно.



С точки зрения аппарата генетических алгоритмов решение по составам партий всех *n* типов представляет собой одну хромосому. Тогда в хромосоме должно быть определено *n* участков, каждый из которых содержит генов (). Геном, образующим часть хромосомы, является информация о количестве данных *i-*го типа в одной из партий. Через  обозначим значения гена (аллель), соответствующее значению элемента матрицы составов партий данных *i-*го типа. В соответствии с условием, что количество генов в участке хромосомы, соответствующем решению по составам партий *i-*го типа, равно , ген  может принимать значение из множества .



В соответствии с базовым подходом использования генетических алгоритмов [6] процесс оптимизации решений предполагает наличие двух стадий: формирование начальной популяции (начальной совокупности решений), изменение популяции (совокупности решений) с целью определения приближенно оптимальных решений. Генерация начальной популяции предполагает формирование некоторого определенного количества хромосом (начальных решений), которые будут изменяться на второй стадии генетического алгоритма. Как было определено выше максимальное количество партий данных каждого типа определяется как , тогда общее количество генов в формируемых хромосомах определяется как . Значения генов () каждого *i-*го участка хромосомы () определяются в соответствии с следующим порядком шагов:



1) значение гена  формируется путем выбора случайным образом одного из значений в множестве (), т.е. первый элемент последовательности предполагается не нулевым;



2) перед формированием значения очередного *h*-го гена (,) выполняется проверка условия ; если данное условие не выполняется, тогда текущее значение *h*-го гена  определяется следующим образом: , после чего формирование *i-*го участка хромосомы (для *i-*го типа данных) прекращается, выполняется переход на шаг 5; при выполнении введенного условия  текущее значение *h*-го гена  формируется путем выбора случайным образом одного из значений в следующем множестве: ;



3) после формирования значения *h*-го гена выполняется проверка условия , означающего невозможность формирования новой партии данных рассматриваемого типа; если данное условие выполняется, тогда текущее значение  *h-*го гена инкрементируется (), осуществляется переход на шаг 5; в случае не выполнения введенного условия реализуется переход к формированию значения  следующего *(h+1)*-го гена – реализуется переход на шаг 2;



4) перед определением значения последнего -го гена выполняется проверка условия , в случае его выполнения значение  последнего -го гена в формируемом участке хромосомы для i-го типа данных определяется следующим образом: , реализуется переход на шаг 5; если введенное условие не выполняется, тогда значение последний сформированного *h-*го гена, такого, что  инкрементируется (), реализуется переход на шаг 5;



5) останов алгоритма формирования участка хромосомы, соответствующего *i-*му типу данных.

Пример реализации приведенного алгоритма для случая () представлен на рисунке 3.1*.* На основе каждой формируемой хромосомы реализуется построение решения , которое в дальнейшем интерпретируется как одно из решений по составам партий данных *n* типов. Таким образом, результатом формирования начальной популяции является полученная совокупность решений, которая в дальнейшем подлежит изменению в соответствии с генетическими операторами.



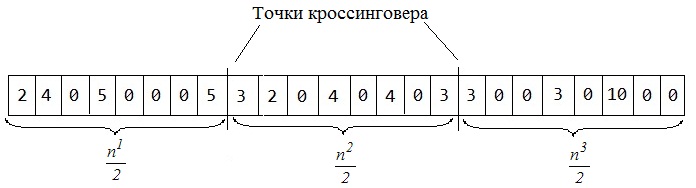


Рисунок 3.1 – Реализация алгоритма формирования хромосомы, соответствующего начальному составу партий

Генетическим операторами, реализуемыми при определении локально оптимальных решений, являются операторы селекции, кроссенговера (скрещивания), мутации и редукции. С целью избежания попадания в «ловушку» локального экстремума применена стохастическая селекция (селекция на основе рулетки). Оператор кроссенговера не предполагает стохастического определения точек скрещивания родительских хромосом для определения их потомков. Точки скрещивания хромосом имеют для рассматриваемой задачи строго заданное положение, определяемое границами участков этих хромосом, соответствующих различным типам данных. Стохастический характер кроссенговера заключается только в определении количества точек скрещивания. В отличии от оператора кроссенговера мутация в рассматриваемой задаче носит целиком стохастический характер и предполагает определение:

1) количества типов данных (соответственно, участков хромосом), для которых будет выполняться мутация генов;

2) местоположение (позиции) и  генов, значения которых и  подлежат мутации.



Под мутацией подразумевается изменение значений ,  генов в позициях  и , которые формируются случайным образом. При этом реализуется увеличение значения  гена в -ой позиции в участке хромосомы, соответствующем *i-*ому типу данных, и уменьшение значения гена в -ой позиции в участке хромосомы, соответствующем этому же типу данных. Оба значения и  изменяются на 2 единицы. Тогда условиями реализации оператора мутации для определяемых случайным образом позиций и  генов со значениями и (соответственно) являются:



1) значение  в позиции  такое, что (т.к. измененное количество данных в партии не может равным  и );



2) значение  в позиции такое, что  и  (т.к. количество данных в партии не может быть отрицательным), (т.к. количество данных в партии не может быть равно 1).



В случае не выполнения какого-либо из условий реализуется повторная генерация значений  и (или) .

Рассмотренный метод определения составов партий данных *i-*ых типов (), реализующий формирование решений в окрестностях разных видов и функционирующий на первом уровне системы, взаимодействует с методом формирования порядков обработки партий на сегментах конвейера (расписаний обработки партий). Введенное ранее предположение об одинаковом порядке обработки партий данных на всех сегментах конвейера позволяет упростить процедуру построения расписаний обработки партий (формируются одинаковые виды последовательностей  () обработки данных на сегментах конвейера). Т.к. количество типов данных ограничено, партия может содержать не менее двух данных каждого типа, порядки обработки данных на сегментах конвейера являются одинаковыми, тогда является ограниченным количество решений по расписаниям обработки партий. Для формирования расписаний обработки партий данных в системе использован метод поиска лучшего решения в рамках окрестностей текущего локального оптимального решения (расписания обработки).



## Анализ эффективности метода формирования составов партий с учетом требования оперативности формирования комплектов из результатов обработки

Проведем анализ работы системы и докажет необходимость метода оптимизации состава партий данных на первом уровне системы. Сравнение необходимо проводить с аналогично построенной системой без модуля оптимизации составов партий данных на первом уровне.

Для сравнения возьмем фиксированные партии – используем начальное решение по составам партий данных всех типов. Фиксированные партии не будут модифицироваться, а сразу отправляться на второй уровень представленной модели.

В работе модуля оптимизации начальное решение улучшается (увеличивается окрестность решения на каждом последующем шаге алгоритма) при использовании метода оптимизации партий данных, описанного выше.

Так же сравним представленный метод с методом генетического алгоритма для формирования партий данных. Алгоритм обоснован и описан в предыдущем разделе.

Проанализируем полученные данные.

Для анализа были взяты несколько промежутков отношений максимальных к минимальным времен обработки и переналадки. Так же были взяты несколько наборов данных количества типов данных, длин конвейера, количества и составов комплектов и времен формирования комплектов.

В качестве временных промежутков были выбраны значение 100 и 150 условных временных единиц для формирования комплектов. Так же введено ограничение на минимальное время обработки и переналадки системы (минимальное время обработки и переналадки равно 2 условным временным единицам).

Для всех временных промежутков установим значение длины конвейера (количества обрабатывающих устройств в системе). Положим это значение равное 5 (в обработке всех требований используется 5 сегментов конвейера). Так же фиксированным является количество типов данных (n равно 5).

Проведем анализ системы при количестве типов комплектов равным 4, количеству комплектов каждого типа равным 2 и максимальному составу комплекта равному 2 единицам требований каждого типа.

Рисунок 4.1 – график зависимости среднего времени формирования комплекта от времени переналадки при фиксированном времени обработки

Рисунок 4.2 – график зависимости среднего времени формирования комплекта от времени переналадки при фиксированном времени обработки

На основании полученных данных можно смело заявить, что при использовании методов оптимизации на первом уровне системы мы получаем лучшее решение, чем решение, основанное на фиксированных партиях и генетическом алгоритме. Алгоритм оптимизации получает оптимальное решение по составу партий данных при условии оперативности формирования комплектов.

## Анализ эффективности метода формирования составов партий при условии формирования комплектов в заданные для них директивные сроки

Проведем анализ работы системы и докажет необходимость метода оптимизации состава партий данных на первом уровне системы. Сравнение необходимо проводить с аналогично построенной системой без модуля оптимизации составов партий данных на первом уровне.

Для сравнения возьмем фиксированные партии – используем начальное решение по составам партий данных всех типов. Фиксированные партии не будут модифицироваться, а сразу отправляться на второй уровень представленной модели.

В работе модуля оптимизации начальное решение улучшается (увеличивается окрестность решения на каждом последующем шаге алгоритма) при использовании метода оптимизации партий данных, описанного выше.

Так же сравним представленный метод с методом генетического алгоритма для формирования партий данных. Алгоритм обоснован и описан в разделе 4.1.

Проанализируем полученные данные.

Для анализа были взяты несколько промежутков отношений максимальных к минимальным времен обработки и переналадки. Так же были взяты несколько наборов данных количества типов данных, длин конвейера, количества и составов комплектов и времен формирования комплектов.

В качестве временных промежутков были выбраны значение 100 и 150 условных временных единиц для формирования комплектов. Так же введено ограничение на минимальное время обработки и переналадки системы (минимальное время обработки и переналадки равно 2 условным временным единицам).

Для всех временных промежутков установим значение длины конвейера (количества обрабатывающих устройств в системе). Положим это значение равное 5 (в обработке всех требований используется 5 сегментов конвейера). Так же фиксированным является количество типов данных (n равно 5).

Проведем анализ системы при количестве типов комплектов равным 4, количеству комплектов каждого типа равным 2 и максимальному составу комплекта равному 2 единицам требований каждого типа.

Рисунок 4.3 – график зависимости суммарного запаздывания формирования комплекта от времени переналадки при фиксированном времени обработки

Рисунок 4.4 – график зависимости суммарного запаздывания времени формирования комплекта от времени переналадки при фиксированном времени обработки

На основании полученных данных можно смело заявить, что при использовании методов оптимизации на первом уровне системы мы получаем лучшее решение, чем решение, основанное на фиксированных партиях и генетическом алгоритме. Алгоритм оптимизации получает оптимальное решение по составу партий данных при условии директивных сроков формирования комплектов.

## Выводы раздела 4

В данном разделе были получены наглядные результаты работы системы, а так же проведен сравнительный анализ и доказательство необходимости использования методов оптимизации решений на первом уровне по отношению к фиксированным партиям данных.

# ЗАКЛЮЧЕНИЕ

В результате выполнения дипломной работы реализован и исследован метод построения расписаний обработки данных разных типов в конвейеризированной системе.

В данной работе рассматриваются алгоритмы формирования эффективных расписаний обработки данных, алгоритмы формирования составов комплектов из результатов обработки партий, алгоритмы формирования эффективных составов партий.

Задача составления расписания является трудновыполнимой, поэтому в данной работе предлагается провести вертикальную декомпозицию обобщенной цели системы и разбить ее на совокупность иерархически упорядоченных уровней (подцелей). При этом достижение каждой из подцелей ведет к выполнению обобщенной цели системы. Особенностью данной задачи является:

1. задаваемые типы данных;
2. задаваемое количество данных каждого определенного типа, что позволяет перейти к формированию партий данных;
3. задаваемой ограничение на длительность реализации.

В ходе работы была описана и обоснована двухуровневая иерархическая теоретико-игровая модель, которая реализует формирование эффективных расписаний обработки партий данных и формирования комплектов из результатов обработки. Обобщенной целью системы является обработка минимизация времен формирования комплектов и минимизация отклонений времен формироавния комплектов от заданных директивных сроков. Каждый уровень модели выполняет следующие действия:

* первый уровень модели формирует эффективный состав партий требований. Критерием оптимальности данного уровня является времена формирования комплектов партий данных. Сформированный состав партий передается на второй уровень;
* второй уровень формирует эффективное расписание обработки партий данных с учетом формирования комплектов.

Для получения оптимального решения вышестоящий уровень системы использует решение, полученное на нижестоящем уровне системы. Для оценки эффективности решения на каждом из уровней используются критерии оценки, которые учитывают:

1. на первом (верхнем) уровне – сумма отклонений времен формирования комплектов;
2. на втором (нижнем) уровне – эффективность использования оборудования конвейерной системы при размещении рассматриваемой партии (добавляемой в расписание ) в последовательностях обработки для формирования комплектов.

На основе теоретико-игровой модели, а так же алгоритмов формирования партий, расписаний и формирования комплектов было реализовано приложение средствами языка С#. Интерфейс данного приложения позволяет задавать все необходимые параметры системы, а на выходе получать эффективное решение для всех уровней иерархии. С помощью этого приложения были проведены исследования эффективности алгоритма формирования эффективного состава партий данных и формирования комплектов.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Кротов К.В. Многоуровневая модель построения расписаний обработки партий данных в конвейерных системах при формировании комплектов и наличии ограничений./ К.В.Кротов// СПб.: Труды СПИИРАН, 2016, Вып. 4(47). – С. 65-91;
2. Mendez C. A. State-of-the-art review of optimization methods for short-term scheduling of batch processes./ C. A. Mendez, J. Cerda, I. E. Grossmann, L. Harjunkoski, M. Fahl. // Computers and Chemical Engineering, 2006, #30.– pp. 913–946.
3. Ковалев М.М. Модели и методы календарного планирования. Курс лекций./ М.М. Ковалев. Минск: Изд-во БГУ, 2004.– 63 с.
4. Tan Y. Comparative Study of Different Approaches to Solve Batch Process Sheduling and Optimisation Problems./ Tan Y., Huangi W., Sun Y., Yue Y.// Proceedings of the 18th International Conference on Automation & Computing, Loughborough University, Leicestershire, UK, 8 September 2012.– pp. 424–444.
5. Adonyi R. Incorporating heat integration in batch process scheduling./ R. Adonyi, J. Romero, L. Puigjaner, F. Friedler.// Applied Thermal Engineering, 2003, # 23.– pp. 1743–1762;
6. Kreipl S. Planning and Scheduling in supply chains: An Overview of Issues in Practice./ S. Kreipl, M. Pinedo.// Production and Operations Management, 2004, vol.17, #1.– pp. 77-92;
7. Steiner G. Minimizing the weight number of late jobs with Batch setup times and delivery costs on a single machine./ G. Steiner, Zhang R.// Mulltiprocessor scheduling: Theory and Applications, Book edited by E. Levner. – Vienna, Austria: Itech Education and Published, 2007.– 436 pp.
8. Koehler F. Optimal Batch Schedules for Parrallel Machines./ F. Koehler, S. Khuller // Algorithms and Data Structures: 13th International Symposium.– Berlin: Springer – Verlag, 2013.– pp. 475– 486;
9. Yugma C. Batching and Scheduling Algoritm for the diffusion Area in Semicondactor Manufacturing./ C. Yugma, S. Davsere-Perez, C. Artiques, O. Sibille.// International Journal of Production Research, Taylor&Francis, 2012, #8.– pp.2118–2132;
10. Chandra P. Managing Batch Processors to reduce lead time in a semiconductor packaging line/ P. Chandra, S. Gupta// International Jornal of Production Research, 1997, #35 (3)– pp. 611– 633;
11. Surjandari I. The Batch Sheduling Model for Dynamic multiitem, Multilevel Production in an assembly Job-Shop with Parrallel Machines./ I. Surjandari, A. Rachman, A. Dhini// International Journal of Technology, 2015,#1.– pp. 84-96;
12. Кротов К.В. Использование аппарата генетических алгоритмов при формировании решений по составам партий данных в двухуровневой задаче построения комплексных расписаний их обработки. // Автоматизированные технологии и производства. Международный научно-технический журнал.– №2, 2017.– С.23-34.
13. .Петросян Л.А. Теория игр./ Л.А.Петросян, Н.А.Зенкевич, Е.А.Семина. – М.: Изд-во «Высшая школа», 1999. – 300с.
14. Гермейер Ю.Б. Игры с непротивоположными интересами / Ю.Б.Гермейер. – М.: Наука, 1976. – 327 с.
15. Месарович, М. Теория иерархических многоуровневых систем [Текст] / М. Месарович, Д. Мако, И. Такахара – М.: Наука, 1973. – 340 с.
16. Топорков, В.В. Модели распределенных вычислений [Текст] / В.В. Топорков. – М.: Физматлит, 2004. – 320 с.
17. Танаев, В.С. Теория расписаний. Групповые технологии [Текст] / В.С. Танаев, М.Я. Ковалёв, Я.М. Шафранский. – Минск: ИТК НАН Беларуси, 1998. – 290 с.
18. Танаев, В.С. Введение в теорию расписаний [Текст] / В.С. Танаев, В.В. Шкурба. – М.: Наука, 1975. – 256 с.
19. Кротов, К.В. Градиентный метод составления расписаний в многостадийной системе с различными временами поступления требований [Текст]. / К.В. Кротов // Вісник СевНТУ «Автоматизация процесів та управління»: збірник наукових праць. – 2010 – Вип. 108. – С. 145-151.
20. Лазарев, А.А. Теория расписаний. Задачи и алгоритмы [Текст] /А.А. Лазарев, Е.Р Гафаров. – М.: МГУ им. М.В. Ломоносова, 2011. – 222 с.
21. Танаев, В.С. Теория расписаний. Многостадийный системы [Текст] / В.С. Танаев, Ю.Н. Сотсков, В.А. Струсевич. – М.: Наука, 1989. – 328 с.
22. Конвей, Р. В. Теория расписаний [Текст] / Р.В. Конвей, В.Л. Максвелл, Л.В. Миллер. – М.: Наука., 1989. – 360 с.
23. Коффман, Э.Г. Теория расписаний и вычислительные машины [Текст] / Э.Г. Коффман : пер. с англ. В.М. Амочкина. – М.: Наука, 1984. – 336 с.
24. Петросян, Л.А. Теория игр. [Текст]: Учебное пособие / Л.А. Петросян, Н.А. Зенкевич, Е.А. Семина. – М.: Высшая школа, 1998. – 304 с. : ил.
25. 25. Новиков Д.А. Теория управления организационными системами [Текст] / Д. А. Новиков. – М.: МПСИ, 2005. – 584 с.
26. 26. Шеллинг Т. Стратегии конфликта [Текст] / Томас Шеллинг : пер. с англ. Т. Даниловой – М.: ИРИСЭН, 2007. – 376 с.
27. 27. Шлее, М. Qt 4.5. Профессиональное программирование на C++ [Текст]. / Макс Шлее : пер. с англ. – Спб.: БХВ-Петербург, 2010. – 885 c.
28. 28. Гонсалес, Р. Цифровая обработка изображений [Текст]. / Р. Гонсалес, Р. Вудс : пер. с англ. В.В. Чепыжова. – М.: Техносфера, 2005. – 1072 с.
29. 29. Джианпинг Жоу (Jianping Zhou). Как получить максимум от конвейерной обработки изображений [Текст] / Жоу Джианпинг : пер. с англ. // Бюллетень научно-технической информации «Компоненты TI. Полный спектр применений». – 2010. – №2(26). – С. 26-30.
30. 30. Авшаров, Е.М. Обработка медицинских изображений как необходимый инструментарий медицинского диагностического процесса [Текст] / Е.М. Авшаров, М.Г. Абгарян, С.А. Сафарянц. – М.: ООО «КУРС-АС1», 2011. – 11с.
31. 31. Banlman I. Handbook of Medical Imaging: Processing and Analysis [Text] / I.Bankman(Ed.). – SanDiego,CA: AcademicPress, 2009 –901 p.

# ПРИЛОЖЕНИЕ А

Листинг программы формирования партий данных и расписаний их обработки и формирования комплектов при условии оперативности и директивных сроков.

Листинг модуля формирования составов партий

using System.Collections.Generic;

using System.Linq;

using System.Windows.Forms;

using System.IO;

namespace newAlgorithm

{

class FirstLevel

{

private readonly List<int> \_i; // Вектор интерпритируемых типов данных

private List<List<int>> \_ai; // Буферизированная матрица составов партий требований на k+1 шаге

private List<List<int>> \_abuf; // Буферизированная матрица составов партий требований на k+1 шаге

private List<List<List<int>>> \_a1; // Матрица составов партий требований на k+1 шаге

private List<List<List<int>>> \_a2; // Матрица составов партий требований фиксированного типа

public List<List<int>> \_a { get; private set; } // Матрица составов партий требований на k шаге

private readonly int \_countType; // Количество типов

private readonly List<int> \_countClaims; // Начальное количество требований для каждого типа данных

private int \_f1; // Критерий текущего решения для всех типов

private int \_f1Buf; // Критерий текущего решения для всех типов

private readonly bool \_staticSolution; // Признак фиксированных партий

private List<int> \_nTemp;

private bool \_typeSolutionFlag;

/// <summary>

/// Конструктор с параметрами

/// </summary>

/// <param name="countType">количество типов рассматриваемых данных</param>

/// <param name="count\_claims">количество требований всех типов данных</param>

public FirstLevel(int countType, List<int> countClaims, bool stat)

{

\_countType = countType;

\_countClaims = countClaims;

\_staticSolution = stat;

\_i = new List<int>(\_countType);

}

/// <summary>

/// Функция копирования значений между матрицами, предотвращающая копирование указателей

/// </summary>

/// <param name="inMatrix">Входная матрица</param>

/// <returns>Выходная матрица</returns>

private List<List<int>> CopyMatrix(IEnumerable<List<int>> inMatrix)

{

return inMatrix.Select(CopyVector).ToList();

}

/// <summary>

/// Функция копирования значений между векторами, предотвращающая копирование указателей

/// </summary>

/// <param name="inMatrix">Входной вектор</param>

/// <returns>Выходной вектор</returns>

private static List<int> CopyVector(List<int> inMatrix)

{

return inMatrix.ToList();

}

/// <summary>

/// Алгоритм формирования начальных решений по составам партий всех типов

/// </summary>

public void GenerateStartSolution()

{

const int claim = 2;

\_a = new List<List<int>>();

for (var i = 0; i < \_countType; i++)

{

\_i.Add(1);

\_a.Add(new List<int>());

\_a[i].Add(\_countClaims[i] - claim);

\_a[i].Add(claim);

}

for (var i = 0; i < \_countType; i++)

{

if (\_a[i][0] < 2 || \_a[i][0] < \_a[i][1])

{

\_a[i].Clear();

\_a[i].Add(\_countClaims[i]);

\_i[i] = 0;

}

}

}

/// <summary>

/// Функция вычисления f1 критерия

/// </summary>

/// <param name="inMatrix">Матрица А на текущем шаге</param>

/// <returns>Значение критериия</returns>

public int GetCriterion(List<List<int>> inMatrix)

{

return inMatrix.SelectMany(t => t).Sum();

}

/// <summary>

/// Функция проверки наличия оставшихся в рассмотрении типов

/// </summary>

/// <param name="type">список всех рассматриваемых типов</param>

/// <returns>наличие еще рассматриваемых типов</returns>

private bool CheckType(IReadOnlyList<int> type)

{

var count = 0;

for (var j = 0; j < \_countType; j++)

{

if (type[j] > 0)

count++;

}

return count != 0;

}

/// <summary>

/// Построчное формирование матрицы промежуточного решени

/// </summary>

/// <param name="type">тип рассматриваемого решения</param>

/// <param name="ind2">индекс подставляемого решения</param>

/// <returns>матрица А с подставленным новым решением в соответствующий тип</returns>

private List<List<int>> SetTempAFromA2(int type, int ind2)

{

var result = CopyMatrix(\_a);

if (ind2 < \_a2[type].Count)

result[type] = CopyVector(\_a2[type][ind2]);

return result;

}

/// <summary>

/// Функция получения неповторяющихся решений в матрице А2 на шаге 9

/// </summary>

/// <param name="inMatrix">входная матрица сформированных решений</param>

/// <returns>Новые решения без повторений</returns>

public List<List<int>> SortedMatrix(List<List<int>> inMatrix)

{

var temp = CopyMatrix(inMatrix);

//Удаление повторяющихся строк

var countLoops = 0;

while (true)

{

for (var i = 1; i < temp.Count; i++)

{

var lastIndexForDelete = temp.FindLastIndex(delegate(List<int> inList)

{

if (inList.Count != temp[i].Count)

{

return false;

}

var countFind = inList.Where((t, k) => t == temp[i][k]).Count();

return countFind == inList.Count ? true : false;

});

if (lastIndexForDelete == i) continue;

temp.RemoveAt(lastIndexForDelete);

inMatrix.RemoveAt(lastIndexForDelete);

}

countLoops++;

if (countLoops > 100)

break;

}

return inMatrix;

}

/// <summary>

/// Удаление повторений новых решений совпадающих с A1

/// </summary>

/// <param name="inMatrix">матрица новых решений</param>

/// <param name="type">рассматриваемый тип</param>

/// <returns>Полученные новые решения</returns>

private List<List<int>> CheckMatrix(List<List<int>> inMatrix, int type)

{

foreach (var row2 in \_a1[type])

{

foreach (var rowMatrix in inMatrix.ToList())

{

if (rowMatrix.Zip(row2, (a, b) => new { a, b }).All(pair => pair.a == pair.b))

{

inMatrix.Remove(rowMatrix);

}

}

}

return inMatrix;

}

/// <summary>

/// Формирование новых решений по составим партий текущего типа данных

/// </summary>

/// <param name="type">рассматриваемый тип</param>

/// <returns>новые решения для этого типа</returns>

private List<List<int>> NewData(int type)

{

var result = new List<List<int>>();

foreach(var row in \_a1[type])

{

for (var j = 1; j < row.Count; j++)

{

result.Add(CopyVector(row));

if (row[0] <= row[j] + 1) continue;

result[result.Count - 1][0]--;

result[result.Count - 1][j]++;

}

if (result[result.Count - 1][0] != row[0]) continue;

{

var summ = row[0];

result[result.Count - 1].Add(2);

for (var j = 1; j < row.Count; j++)

{

summ += row[j];

result[result.Count - 1][j] = 2;

}

result[result.Count - 1][0] = summ - 2 \* (result[result.Count - 1].Count - 1);

}

}

var count = 0;

while (true)

{

for (var i = 1; i < result.Count; i++)

{

for (var j = 1; j < result[i].Count; j++)

{

if (result[i][j] <= result[i][j - 1]) continue;

result.Remove(result[i]);

break;

}

}

count++;

if (count > 3)

{

break;

}

}

result = SortedMatrix(result);

result = CheckMatrix(result,type);

return result;

}

/// <summary>

/// Формирование новых решений по составим партий текущего типа данных

/// </summary>

/// <param name="m">матрица А для печати</param>

/// <returns>строка с составами партий по типам</returns>

private static string PrintA(IEnumerable<List<int>> m)

{

var result = "";

foreach (var t in m)

{

for (var j = 0; j < t.Count - 1; j++)

{

result += t[j] + ", ";

}

result += t[t.Count - 1] + "; ";

}

return result;

}

/// <summary>

/// НУжна для отладки вывода массива

/// </summary>

/// <param name="m">входной лист</param>

/// <returns>лист в виде строки</returns>

private static string PrintList(List<int> m)

{

var result = "";

foreach (var t in m)

{

result += t + ", ";

}

return result;

}

/// <summary>

/// Проверка на достижение максимально возможного решения по составам типов

/// </summary>

/// <param name="inMatrix">Матрица текущих составов</param>

private void CheckSolution(IReadOnlyList<List<int>> inMatrix)

{

for (var i = 0; i < inMatrix.Count; i++)

{

var elem = inMatrix[i][0];

if (elem != 2) continue;

var count = 1;

for (var j = 1; j < inMatrix[i].Count; j++)

{

if (inMatrix[i][j] == elem)

{

count++;

}

}

if (count == inMatrix[i].Count)

{

\_i[i] = 0;

}

}

}

/// <summary>

/// Алгоритм формирования решения по составам паритй всех типов данных

/// </summary>

public void GenetateSolutionForAllTypes(string fileName)

{

using (var file = new StreamWriter(fileName))

{

GenerateStartSolution();

var shedule = new Shedule(\_a);

shedule.ConstructShedule();

\_f1 = shedule.GetTime();

//MessageBox.Show(PrintA(A) + " Время обработки " + f1);

\_f1Buf = \_f1;

file.WriteLine(\_f1Buf);

var maxA = CopyMatrix(\_a);

\_typeSolutionFlag = true;

if (!\_staticSolution)

{

while (CheckType(\_i))

{

// Буферезируем текущее решение для построение нового на его основе

\_ai = CopyMatrix(\_a);

if (\_typeSolutionFlag)

{

\_a1 = new List<List<List<int>>>();

for (var i = 0; i < \_countType; i++)

{

\_a1.Add(new List<List<int>>());

\_a1[i].Add(new List<int>());

\_a1[i][0] = CopyVector(\_a[i]);

}

\_typeSolutionFlag = false;

}

var tempA = CopyMatrix(\_ai);

\_abuf = CopyMatrix(\_ai);

\_f1Buf = \_f1;

// Для каждого типа и каждого решения в типе строим новое решение и проверяем его на критерий

\_a2 = new List<List<List<int>>>();

string s;

//file.WriteLine("окрестность 1 вида");

for (var i = 0; i < \_countType; i++)

{

\_a2.Add(new List<List<int>>());

if (\_i[i] <= 0) continue;

\_a2[i] = NewData(i);

for (var j = 0; j < \_a2[i].Count; j++)

{

tempA = SetTempAFromA2(i, j);

shedule = new Shedule(tempA);

shedule.ConstructShedule();

var fBuf = shedule.GetTime();

s = PrintA(tempA);

//file.Write(s + " " + fBuf);

//MessageBox.Show(s + " Время обработки " + fBuf);

if (fBuf < \_f1Buf)

{

\_abuf = CopyMatrix(tempA);

\_typeSolutionFlag = true;

\_f1Buf = fBuf;

//file.Write(" +");

}

//file.WriteLine();

}

}

if (!\_typeSolutionFlag)

{

//file.WriteLine("комбинации типов");

for (var i = 0; i < \_countType - 1; i++)

{

if (\_i[i] <= 0) continue;

for (var j = i + 1; j < \_countType; j++)

{

if (\_i[j] <= 0) continue;

\_a2[i] = NewData(i);

\_a2[j] = NewData(j);

for (var ii = 0; ii < \_a2[i].Count; ii++)

{

for (var jj = 0; jj < \_a2[j].Count; jj++)

{

{

tempA = SetTempAFromA2(i, ii);

tempA[j] = CopyVector(SetTempAFromA2(j, jj)[j]);

shedule = new Shedule(tempA);

shedule.ConstructShedule();

var fBuf = shedule.GetTime();

s = PrintA(tempA);

//file.Write(s + " " + fBuf);

//MessageBox.Show(s + " Время обработки " + fBuf);

if (fBuf < \_f1Buf)

{

\_abuf = CopyMatrix(tempA);

\_typeSolutionFlag = true;

\_f1Buf = fBuf;

//file.Write(" +");

}

//file.WriteLine();

}

}

}

}

}

}

if (\_typeSolutionFlag)

{

//MessageBox.Show("Лучшее решение "+PrintA(Abuf) + " Время обработки " + f1Buf);

\_a = CopyMatrix(\_abuf);

\_f1 = \_f1Buf;

}

else

{

for (int i = 0; i < \_countType; i++)

{

\_a1[i] = CopyMatrix(\_a2[i]);

if (!\_a1[i].Any() || !\_a1[i][0].Any())

{

\_i[i] = 0;

}

}

}

}

}

file.WriteLine(\_f1);

file.Close();

//MessageBox.Show("Решения найдены");

}

}

/// <summary>

/// Формирование перебора для всех возможных решений из А2

/// </summary>

/// <param name="ind">текущий индекс изменяемого решения для 1 типа</param>

/// <param name="\_n">Матрица номеров решений из А2</param>

/// <param name="f">Файл для записей логов</param>

///Менят здесь для \_\\*РУСЛАН\*/\_

private void GenerateCombination(int ind, List<int> \_n)

{

for (int i = \_countType - 1; i >= 0; i--)

{

for (int j = 0;j <\_a2[i].Count; j++)

{

\_n[i]=j;

//f.WriteLine(PrintList(\_n));

GetSolution(\_n);

}

}

}

/// <summary>

/// Подстановка данных из перебора и вычисление решения

/// </summary>

/// <param name="\_n">Массив индексов решений из А2</param>

/// <param name="f">Файл для записей логов</param>

private void GetSolution(List<int> \_n)

{

var tempA = CopyMatrix(\_a);

for (var j = 0; j < \_countType; j++)

{

if (\_n[j] >= 0)

{

tempA[j] = CopyVector(SetTempAFromA2(j, \_n[j])[j]);

}

}

var shedule = new Shedule(tempA);

shedule.ConstructShedule();

var r = shedule.RetyrnR();

var sets = new Sets(Form1.compositionSets, Form1.timeSets);

sets.GetSolution(r);

var time = sets.GetNewCriterion(Form1.direct);

var s = PrintA(tempA);

//f.Write(s + " - " + fBuf);

//MessageBox.Show(s + " Время обработки " + fBuf);

if (time < \_f1Buf)

{

\_abuf = CopyMatrix(tempA);

\_typeSolutionFlag = true;

\_f1Buf = time;

//file.Write(" +");

}

//f.WriteLine();

}

/// <summary>

/// Алгоритм формирования решения по составам паритй всех типов данных

/// </summary>

public int[] GenetateSolutionForAllTypesSecondAlgorithm()

{

var sets = new Sets(Form1.compositionSets, Form1.timeSets);

var result = new[] { 0, 0 };

//using (var f = new StreamWriter("standartOutData.txt", true))

{

GenerateStartSolution();

var shedule = new Shedule(\_a);

shedule.ConstructShedule();

var r = shedule.RetyrnR();

sets.GetSolution(r);

var time = sets.GetNewCriterion(Form1.direct);

var \_f1 = time;

\_f1Buf = \_f1;

result[0] = \_f1Buf;

var maxA = CopyMatrix(\_a);

\_typeSolutionFlag = true;

if (!\_staticSolution)

{

while (CheckType(\_i))

{

// Буферезируем текущее решение для построение нового на его основе

\_ai = CopyMatrix(\_a);

if (\_typeSolutionFlag)

{

\_a1 = new List<List<List<int>>>();

for (var i = 0; i < \_countType; i++)

{

\_a1.Add(new List<List<int>>());

\_a1[i].Add(new List<int>());

\_a1[i][0] = CopyVector(\_a[i]);

}

\_typeSolutionFlag = false;

}

var tempA = CopyMatrix(\_ai);

\_abuf = CopyMatrix(\_ai);

\_f1Buf = \_f1;

// Для каждого типа и каждого решения в типе строим новое решение и проверяем его на критерий

\_a2 = new List<List<List<int>>>();

string s;

for (var i = 0; i < \_countType; i++)

{

\_a2.Add(new List<List<int>>());

if (\_i[i] <= 0) continue;

\_a2[i] = NewData(i);

for (var j = 0; j < \_a2[i].Count; j++)

{

tempA = SetTempAFromA2(i, j);

shedule = new Shedule(tempA);

shedule.ConstructShedule();

r = shedule.RetyrnR();

sets = new Sets(Form1.compositionSets, Form1.timeSets);

sets.GetSolution(r);

time = sets.GetNewCriterion(Form1.direct);

s = PrintA(tempA);

//f.Write(s + " - " + time);

if (time < \_f1Buf)

{

\_abuf = CopyMatrix(tempA);

\_typeSolutionFlag = true;

\_f1Buf = time;

}

//f.WriteLine();

}

}

if (!\_typeSolutionFlag)

{

List<int> \_n = new List<int>();

\_nTemp = new List<int>();

for (int i = 0; i < \_countType; i++)

{

\_nTemp.Add(0);

\_n.Add(\_a2[i].Count);

if (\_n[i] == 0) \_n[i] = -1;

}

GenerateCombination(0, \_nTemp);

}

if (\_typeSolutionFlag)

{

\_a = CopyMatrix(\_abuf);

\_f1 = \_f1Buf;

}

else

{

for (int i = 0; i < \_countType; i++)

{

\_a1[i] = CopyMatrix(\_a2[i]);

if (!\_a1[i].Any() || !\_a1[i][0].Any())

{

\_i[i] = 0;

}

}

}

//f.WriteLine("------------------");

}

}

result[1] = \_f1;

//f.Close();

}

return result;

}

}

}

Листинг модуля расписания

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace newAlgorithm

{

class Shedule

{

private List<List<int>> \_r;

public static List<List<int>> Treatment;

public static List<List<List<int>>> Switching;

private int \_timeConstructShedule;

public static int L;

private List<List<List<int>>> \_startProcessing;

private List<List<List<int>>> \_endProcessing;

private List<SheduleElement> \_rWithTime;

public Shedule(List<List<int>> r, int l)

{

this.\_r = r;

L = l;

}

/// <summary>

/// Формирование матрицы для передачи её в модуль расписания

/// </summary>

/// <param name="m">входная матрица А</param>

/// <returns>сформированная матрица для уровня расписания</returns>

private List<List<int>> GenerateR(IReadOnlyList<List<int>> m)

{

var result = new List<List<int>>();

var summ = m.Sum(t => t.Count);

var maxColumn = 0;

for (var j = 0; j < summ; j++)

{

result.Add(new List<int>());

for (var i = 0; i < m.Count; i++)

{

result[j].Add(0);

}

}

for (var i = 0; i < m.Count; i++)

{

if (m[i].Count > maxColumn)

{

maxColumn = m[i].Count;

}

}

var ind = 0;

for (var j = 0; j < maxColumn; j++)

{

for (var i = 0; i < m.Count; i++)

{

if (m[i].Count > j)

{

result[ind][i] = m[i][j];

ind++;

}

}

}

return result;

}

public List<SheduleElement> RetyrnR()

{

\_rWithTime = new List<SheduleElement>();

for (int i = 0; i < \_endProcessing[\_endProcessing.Count - 1].Count; i++)

{

var ind = ReturnRIndex(i);

\_rWithTime.Add(new SheduleElement(\_r[i][ind], ind, \_endProcessing[\_endProcessing.Count - 1][i]));

}

return \_rWithTime;

}

public Shedule(List<List<int>> r)

{

\_r = GenerateR(r);

}

private void CalculateShedule()

{

\_startProcessing = new List<List<List<int>>>();

\_endProcessing = new List<List<List<int>>>();

for (var i = 0; i < L; i++)//количество приборов

{

\_startProcessing.Add(new List<List<int>>());

\_endProcessing.Add(new List<List<int>>());

for (var k = 0; k < \_r.Count; k++)//количество партий

{

var ind = ReturnRIndex(k);

var elem = \_r[k][ind];

if (elem == -1)

{

elem = 1;

}

\_startProcessing[i].Add(new List<int>());

\_endProcessing[i].Add(new List<int>());

for (var p = 0; p < elem; p++)//количество требований

{

\_startProcessing[i][k].Add(0);

\_endProcessing[i][k].Add(0);

}

}

}

/\*\*/

var yy = 0;

var zz = 0;

var xx = 0;

for (var i = 0; i < L; i++)

{

for (var j = 0; j < \_r.Count; j++)

{

var index = ReturnRIndex(j);

for (var k = 0; k < \_r[j][index]; k++)

{

var timeToSwitch = (index == xx && j != 0) ? 0 : Switching[0][xx][index];

if (i > 0)

{

\_startProcessing[i][j][k] = Math.Max(\_endProcessing[i][yy][zz] + timeToSwitch, \_endProcessing[i - 1][j][k]);

}

else

{

\_startProcessing[i][j][k] = \_endProcessing[i][yy][zz] + timeToSwitch;

}

\_endProcessing[i][j][k] = \_startProcessing[i][j][k] + Treatment[i][index];

\_timeConstructShedule = \_endProcessing[i][j][k];

yy = j;

zz = k;

xx = index;

}

}

yy = 0;

zz = 0;

xx = 0;

}

/\*\*/

/\*

for (var j = 0; j < \_r[0].Count; j++)

{

var index = ReturnRIndex(j);

var type = j == 0 ? index : ReturnRIndex(j - 1);

var timeToSwitch = (type == index && j != 0) ? 0 : Switching[0][type][index];

var timeToTreament = Treatment[0][index];

if (j > 0)

{

var last = \_r[type][j - 1] - 1;

\_startProcessing[0][j][0] = \_endProcessing[0][j - 1][last];

}

for (var k = 0; k < \_r[index][j]; k++)

{

if (k == 0)

{

\_startProcessing[0][j][k] += timeToSwitch;

}

else

{

\_startProcessing[0][j][k] += \_endProcessing[0][j][k - 1];

}

\_endProcessing[0][j][k] = \_startProcessing[0][j][k] + timeToTreament;

}

}

for (var i = 1; i < L; i++)

{

for (var j = 0; j < \_r[0].Count; j++)

{

var index = ReturnRIndex(j);

var type = j == 0 ? index : ReturnRIndex(j - 1);

var timeToSwitch = (type == index && j > 0) ? 0 : Switching[i][type][index];

var timeToTreament = Treatment[i][index];

for (var k = 0; k < \_r[index][j]; k++)

{

if (k == 0)

{

if (j > 0)

{

var last = \_r[type][j - 1] - 1;

\_startProcessing[i][j][k] += Math.Max(\_endProcessing[i][j - 1][last] + timeToSwitch,

\_endProcessing[i - 1][j][k]);

}

else

{

\_startProcessing[i][j][k] += \_endProcessing[i - 1][j][k] + timeToSwitch;

}

}

else

{

\_startProcessing[i][j][k] += Math.Max(\_endProcessing[i][j][k - 1], \_endProcessing[i - 1][j][k]);

}

\_endProcessing[i][j][k] = \_startProcessing[i][j][k] + timeToTreament;

\_timeConstructShedule = \_endProcessing[0][j][k];

}

}

}\*/

}

public int ReturnRIndex(int j)

{

for (var i = 0; i < \_r[j].Count; i++)

{

if (\_r[j][i] > 0)

return i;

}

return -1;

}

private static List<List<int>> CopyMatrix(IReadOnlyList<List<int>> inMatrix)

{

var ret = new List<List<int>>();

for (var i = 0; i < inMatrix.Count; i++)

{

ret.Add(new List<int>());

for (var j = 0; j < inMatrix[i].Count; j++)

{

ret[i].Add(inMatrix[i][j]);

}

}

return ret;

}

private void ChangeColum(int ind1, int ind2)

{

var indd1 = 0;

var indd2 = 0;

for (var i = 0; i < \_r[ind1].Count; i++)

{

if (\_r[ind1][i] > 0)

{

indd1 = i;

}

}

for (var i = 0; i < \_r[ind2].Count; i++)

{

if (\_r[ind2][i] > 0)

{

indd2 = i;

}

}

var temp = \_r[ind1][indd1];

\_r[ind1][indd1] = 0;

\_r[ind1][indd2] = \_r[ind2][indd2];

\_r[ind2][indd2] = 0;

\_r[ind2][indd1] = temp;

}

public List<List<int>> ConstructShedule()

{

var tempTime = 9999999;

CalculateShedule();

var tempR = CopyMatrix(\_r);

tempTime = \_timeConstructShedule;

for (var i = 0; i < \_r.Count - 1; i++)

{

for (var j = i + 1; j < \_r.Count; j++)

{

ChangeColum(i, j);

CalculateShedule();

if (tempTime >= \_timeConstructShedule) continue;

\_r = tempR;

\_timeConstructShedule = tempTime;

}

}

return \_r;

}

public int GetTimeWithCriterium(int tz, out int crit)

{

var criterier = 0;

ConstructShedule();

for (int numberPocess = 0; numberPocess < L; numberPocess++)

{

for (int numberBatch = 0; numberBatch < \_startProcessing[numberPocess].Count; numberBatch++)

{

for (int numberWork = 0; numberWork < \_startProcessing[numberPocess][numberBatch].Count; numberWork++)

{

criterier += \_endProcessing[numberPocess][numberBatch][numberWork] - \_startProcessing[numberPocess][numberBatch][numberWork];

}

}

criterier -= \_startProcessing[numberPocess][0][0];

}

crit = (tz \* L) - criterier;

return \_timeConstructShedule;

}

public int GetTime()

{

ConstructShedule();

return \_timeConstructShedule;

}

}

}

Листинг модуля формирования комплектов

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace newAlgorithm

{

class Sets

{

private readonly int \_types;

private readonly List<List<int>> \_composition;

private readonly List<List<int>> \_time;

private List<List<Kit>> \_readySets;

/// <summary>

///

/// </summary>

/// <param name="countType"></param>

/// <param name="composition"></param>

/// <param name="time"></param>

public Sets(List<List<int>> composition, List<List<int>> time)

{

\_types = composition.Count;

\_composition = composition;

\_time = time;

\_readySets = new List<List<Kit>>();

for (int i = 0; i < \_types; i++)

{

\_readySets.Add(new List<Kit>());

for (int j = 0; j < time[i].Count; j++)

{

\_readySets[i].Add(new Kit(composition[i], time[i][j]));

}

}

}

/// <summary>

/// Новый критерий

/// </summary>

/// <returns></returns>

public int GetNewCriterion(bool direct)

{

if (direct)

{

int res = 0;

foreach (var row in \_readySets)

{

foreach (var elem in row)

{

if (elem.GetTime() > elem.GetCompositionTime())

{

res += elem.GetTime() - elem.GetCompositionTime();

}

}

}

return res;

}

else

{

int res = 0;

var count = 0;

foreach (var row in \_readySets)

{

count += row.Count;

foreach (var elem in row)

{

res += elem.GetTime();

if (res < elem.GetTime())

{

res = elem.GetTime();

}

}

}

return (int)(res / count);

}

}

/// <summary>

/// Старый критерий

/// </summary>

/// <returns></returns>

public int GetCriterion()

{

int res = 0;

foreach (var row in \_readySets)

{

foreach (var elem in row)

{

if (res < elem.GetTime())

{

res = elem.GetTime();

}

}

}

return res;

}

/// <summary>

///

/// </summary>

/// <param name="batch"></param>

/// <param name="type"></param>

/// <param name="time"></param>

/// <returns></returns>

protected void AddBatches(SheduleElement sheduleElement)

{

var sets = new List<Kit>();

foreach (var row in \_readySets)

{

foreach (var elem in row)

{

sets.Add(elem);

}

}

sets.Sort(

(Kit kit1, Kit kit2) => kit1.CompareTo(kit2)

);

foreach (var elem in sets)

{

if (!elem.IsSetAllComposition())

{

sheduleElement = elem.AddBatch(sheduleElement.getValue(), sheduleElement.getType(), sheduleElement.getTime());

}

if (sheduleElement.getValue() <= 0)

{

return;

}

}

}

/// <summary>

///

/// </summary>

/// <param name="shedule"></param>

public void GetSolution(List<SheduleElement> shedule)

{

foreach (var element in shedule)

{

AddBatches(element);

}

}

}

}

# ПРИЛОЖЕНИЕ Б

Графики прогонов тестовых данных через систему для удостоверения работоспособности системы в целом и доказательства эффективности использования предложенной системы.

Рисунок Б.1 - график зависимости среднего времени формирования комплекта от времени переналадки при фиксированном времени обработки

Рисунок Б.2 - график зависимости среднего времени формирования комплекта от времени переналадки при фиксированном времени обработки

Рисунок Б.3 - график зависимости суммарного запаздывания времени формирования комплекта от времени переналадки при фиксированном времени обработки

Рисунок Б.4 - график зависимости суммарного запаздывания времени формирования комплекта от времени переналадки при фиксированном времени обработки