Содержание

Введение

Современные системы автоматизации производств глубоко интегрируются в реальные процессы. При этом интегрируются на уровне SCADA (Supervisory Control And Data Acquisition — диспетчерское управление и сбор данных) систем, на уровне сбора первичных данных, если система производства слабо автоматизирована, то интеграция происходит на уровне носимых устройств, систем распределения задач, а также систем исполнения процессов.

Разрабатываемая система цифрового двойника, глубоко интегрированная в производственные процессы, представляет из себя не что иное, как киберфизическую систему [1] [2], которая на основе обратной связи относительно физического объекта принимает решение, анализирует его поведение и формирует управляющие воздействия. Более того, разрабатываемая система ориентирована на то, чтобы быть встроенным в качестве главного узла принятия решений в системе автоматизированного производства.

Глобальная цель такого рода разработок направлена на то, чтобы исключить менеджмент среднего уровня, заменив его алгоритмами. Оставить только менеджеров высокого уровня, которые будут принимать решения относительно направления развития производства, руководствуясь данными цифровых двойников для того, чтобы принятие решения не сводилось к персональным оценкам и в них была объективность.

Актуальность темы исследования. Создание цифрового двойника производства перспективное направление, как безопасный способ получение желаемого результата от реального объекта, не прибегаю к тестированию на реальном производстве. Методы моделирования и, в частности имитационного, постоянно модернизируются, чтобы достичь максимальной точности по отношению к моделируемым объектам.

Степень теоретической разработанности темы. Так как проблема актуальна, по данной тематике существуют большое количество публикаций. Данные публикации в целом сосредоточены на разработке новых подходов к моделированию производств, но также есть и обзорные исследования. Много решений типовых задач. Конкретная специфика требует специальной проработки.

Целью работы является разработка гибридной имитационной модели производственного планирования, а также методов оптимизации оперативного плана.

Для достижения данной цели были поставлены следующие задачи:

1) обзор систем имитационного моделирования, подходов к реализации имитационной модели;

2) разработка алгоритмов и методов имитационного моделирования;

3) алгоритмы оптимизации оперативного плана на основе имитационной модели;

4) алгоритмы оптимизации планирования конвейеризированных производств.

Область исследования. Областью исследования являются алгоритмы имитационного моделирования производственных процессов, а также область их подходов к оптимизации.

Объект исследования. Объектом исследования является подсистема имитационного моделирования и оптимизации производственных процессов.

Научная новизна. Научная новизна исследования проявляется в гибридном подходе к моделированию, сочетающем аналитические и алгоритмические элементы.

Практическая значимость системы. Результаты данного исследования могут будут применены в системах планирования производственными процессами. Одной из таких систем является Система планирования и прогнозирования, разрабатываемая в лаборатории кибер – физических систем.

Апробация результатов исследования. Результаты работы используются в проектах Системы планирования и прогнозирования. Результаты исследований были представлены на X международной научно-практической конференции молодых ученых «Программная инженерия и компьютерная техника (Майоровские чтения)», на VIII Конгрессе молодых ученых (2019) и XLVIII научная учебно-методическая конференция Университета ИМТО (2019).

Объем и структура работы. Работа содержит четыре главы, первая глава является обзорной, где рассмотрено развитие, функции систем управления и планирования предприятием. Приведен обзор методов моделирования в производстве, а также обзор подходов имитационного моделирования. Во второй главе приведен обзор архитектуры подсистемы имитационного моделирования и оптимизации. В третьей главе детально рассмотрен процесс планирования, а также подходов к оптимизации. В четвертой главе рассмотрен вопрос оптимизации плана работы на такт, приведен обзор и представлены подходы к решению данной задачи.

1 Автоматизация производственного планирования

1.1 Развитие систем управления и планирования предприятием

На сегодняшний день, в условиях серьезной конкуренции очень важно следить за всеми новинками технологического прогресса и своевременно внедрять в структуру производства. Так организация предприятия напрямую влияет на эффективность производства. Основным направлением организацией предприятием в последнее время относят системы ИСУП (информационная система управления проектами), которые позволяют достичь следующих задач: оптимизация производственного процесса, снижение издержек и повышение эффективности производства.

Данные системы начали появляться с развитием компьютерных технологий в начале 80-х годов. Одной из главных причин появления данных систем является нехватка административного, бухгалтерского и технического персонала, который обладал бы достаточной квалификацией для обработки информации предприятия, также стало понятно, что предприятия не могут позволять себе большие объемы материального запаса для производства продукции. Это привело к появлению систем планирования потребности в ресурсах. Первым шагом в этом направлении был MRP (Materials Resource Planning), который включал только материалы планирования для производства [3].

Основной концепция MPR заключается в минимизации затрат, связанных с запасами, а также расчет сколько и в какие сроки необходимо произвести конечный продукт.

Недостатками данной системы является, то, что при расчете потребностей в материалах не учитываются производственные мощности, их загрузки, трудозатраты и т. д.

Логическим продолжением MPR системы стала система MPR 2, которая в отличие от предшественника учитывала финансовую составляющую предприятия, а также охватывала более широкий спектр ресурсов. Это позволило компаниям иметь интегрированную бизнес-

систему, которая выводила требования к материалам и мощности, связанные с желаемым планом операций, позволяла вводить подробные данные о деятельности, переводить все это в финансовый отчет и предложить план действий для решения тех вопросов, которые были не в соответствии с желаемым результатом.

К началу 1990-х годов постоянные улучшения в технологии позволили расширить MRP II, включив в него планирование ресурсов для всего предприятия. Такие области, как дизайн продукта, хранение информации, планирование мощностей, системы связи, управление персоналом, финансы и управление проектами, теперь могут быть включены в план. Отсюда и термин ERP (Enterprise Resource Planning). И ERP можно использовать не только в производственных компаниях, но и в любой компании, которая хочет повысить конкурентоспособность путем наиболее эффективного использования всех своих активов, включая информацию [4].

Разрабатываемое в дипломной работе проект принадлежит к классу ERP-систем. Многие современные ERP-системы разработаны по модульному принципу, поэтому существует возможность выбирать и внедрять только те модули, которые необходимы клиенту [5].

В данной работе рассматривается одна из частей ERP систем, отвечающая за сопоставление конструкторских и технологических спецификаций, определяющих состав конечного продукта и ресурсов предприятия. На основании данного сопоставления построение плана производственного процесса, учитывающие ограничения предприятия, а также реализация частных математических моделей. Математические модели призваны оптимизировать производственный процесс в зависимости от специфики [1] предприятия.

[1]Примером такой специфики является конвейеризированное предприятие.

1.2 Функции систем управления и планирование предприятием

ИСУП способна эффективно поддерживать производство в соответствии с расписанием посредством анализа данных и простой интеграции на предприятии. Хотя система не может самостоятельно управлять производственным оборудованием, она все же способна поддерживать постоянный поток материалов по всей цепочке поставок с помощью возможностей принятия решений. Различные функции системы ИСУП включают в себя следующее:

— высокая точность соблюдения сроков (и поставка заданных количеств);

— оптимальная загрузка производственных мощностей;

— короткие производственные циклы;

— минимальный уровень капиталовложения;

— поддержание необходимого уровня складских запасов и материалов на производстве;

— высокая гибкость;

— минимизация расходов.

В следующем разделе приведены источники роста эффективности, используемые ИСУП для достижения оптимального результата.

1.3 Источники роста эффективности

В промышленности есть огромные неиспользованные резервы роста производительности труда. Они могут быть подразделены на резервы снижения трудоемкости продукции и резервы рабочего времени.(см. рисунок 1.1)

Резервы снижения трудоемкости выявляются и реализуются в виде экономии рабочего времени, затрачиваемого непосредственно на выполнение рабочих операций.

Резервы фонда рабочего времени реализуются путем повышения эффективности использования рабочего процесса для данного коллектива в течение определенного планового периода [6].

Рисунок 1.1 — Резервы роста производительности труда

Исходя из информации представленной на рисунке 1.1, можно сделать вывод, что повышение эффективности производства может быть достигнуто, как путем грамотной организации резервов рабочего времени, так и путем пересмотра техники выполнения рабочих операций, но не все резервы повышения производительности можно решить в рамках ИСУП. К таким резервам относится конструктивные особенности изделия, так как требуют изменения исходной конструкции продукта, что не является задачей ИУС.

1.4 Обзор методов планирования производственных процессов.

В данном разделе приведен перечень используемых методов моделирования в производстве.

Математический

Описание. Составление математического эквивалента процесса или объекта, отражающий его основные свойства.

Область применения. Любые процессы и объекты, поддающиеся математическому описанию.

Достоинства метода. Широкая область применения.

Недостатки метода. Достаточно сложно построить модель, адекватно учитывающую все факторы.

Статистический

Описание. Модель основывается на выявленных статических закономерностях.

Область применения. Процессы, по которым можно собрать массив статических данных.

Достоинства метода. При наличии качественных данных метод точен и, при использовании специализированного ПО, прост в применении.

Недостатки метода. Большие требования к статическим данным и сложность их сбора.

Экономико-математический

Описание. Раздел включает в себя методы для решения экономических задач.

Область применения. Экономические процессы.

Достоинства метода. Метод способен моделировать экономические процессы.

Имитационный

Описание. Изучаемая система заменяется моделью, с достаточной точностью описывающей реальную систему, с ней проводятся эксперименты с целью получения информации.

Область применения. Метод используется когда дорого или невозможно использовать реальную модель и/или аналитическую модель.

Достоинства метода. Создается максимально приближенная модель, можно управлять временем системы и другими её характеристиками.

Недостатки метода. Сложность описания всех условий и высокие требования вычислительной мощности.

Физический

Описание. Экспериментальное моделирование, основанное на физическом подобии уменьшенной в размерах модели.

Область применения. Применяется при невозможности применения аналитического метода или воспроизведения в реальном размере.

Достоинства метода. Область применения, недоступная другим методам.

Недостатки метода. Метод может дать надёжные результаты лишь при соблюдении физического подобия модели.

Для задач моделирования сложных, сборочных производств больше всего подходит метод имитационного моделирования, так как эксперементировать с реальным объектом экономически нецелесообразно, а также невозможно учесть все зависимости, что усложняет процесс аналитического моделирования.

1.5 Обзор подходов к имитационному моделированию

В прошлом производственные инструменты моделирования классифицировались как языки или симуляторы [7]. Языки были очень гибкими инструментами, но довольно сложными в использовании менеджерами и слишком трудоемкими. Симуляторы были более удобными для пользователя, но они шли с довольно жесткими шаблонами, которые недостаточно адаптировались к быстро меняющимся технологиям производства. В настоящее время доступно программное обеспечение, которое сочетает в себе гибкость и удобство для обоих, но все же некоторые авторы сообщают, что использование данного подхода к моделированию для проектирования и оптимизации производственных процессов является не эффективным [8] [9].

Одним из наиболее часто используемых методов является моделирование дискретных событий [10]. Этот тип моделирования позволяет оценить производительность системы путем статистического и вероятностного воспроизведения взаимодействий всех ее компонентов в течение определенного периода времени. В некоторых случаях моделирование производственных систем требует непрерывного подхода к моделированию [11]. Это те случаи, когда состояния системы постоянно меняются, как, например, при движении жидкостей на нефтеперерабатывающих или химических заводах. Поскольку непрерывное моделирование не может быть смоделировано цифровыми компьютерами, оно выполняется небольшими дискретными шагами. Это полезная функция, поскольку во многих случаях необходимо комбинировать как непрерывное, так и дискретное моделирование. Это называется гибридным моделированием [12], которое используется во многих отраслях, например в пищевой промышленности [8].

На данный момент существует большое количество подходов к имитационному моделированию. Ниже приведена таблица 1.1 задачи и подходов моделирования в производстве [13]:

Таблица 1.1 — Задачи и методы моделирования производства

Задача	Подход	Описание области задачи
Балансировка сборочной линии	Дискретно-событийное моделирование(ДСМ)	Проектирование и балансировка сборочной линии
Планирование мощности	Системная динамика(СД), Метод Монте-Карло(МК), ДСМ	Неопределенность из-за изменения уровней мощности, увеличения текущих ресурсов, улучшения текущих операций для увеличения мощности
Управление запасами	ДСМ, МК	Стоимость имущества, уровни запасов, пополнение, определение размеров партии
Just-in-time	ДСМ	Проектирование систем Канбан
Планирование	ДСМ	Пропускная способность, надежность доставки, последовательность операций, планирование производства, минимизация времени простоя, спрос, готовность заказа
Система управления цепями поставок	ДСМ, СД, Агентное моделирование (АГ), Сети Петри (СП),	Нестабильность в цепочке поставок, системах инвентаризации / распределения
Распределение ресурсов	ДСМ	Выделение оборудования для улучшения технологических процессов, сырья для заводов, выбора ресурсов
Планирование производства и управление запасами	ДСМ, АГ,	Страховой запас, размер партии, узкие места, правила прогнозирования и планирования
Прогнозирование	Гибридные технологии	Сравнение разных моделей прогнозирования

Как видно из таблицы 1.1 по результатам исследования [13] было выявлено, что наиболее используемый метод моделирования производсвенного планирования является дискретно-событийная модель.

2 Система планирования производства

2.1 Архитектура системы планирования

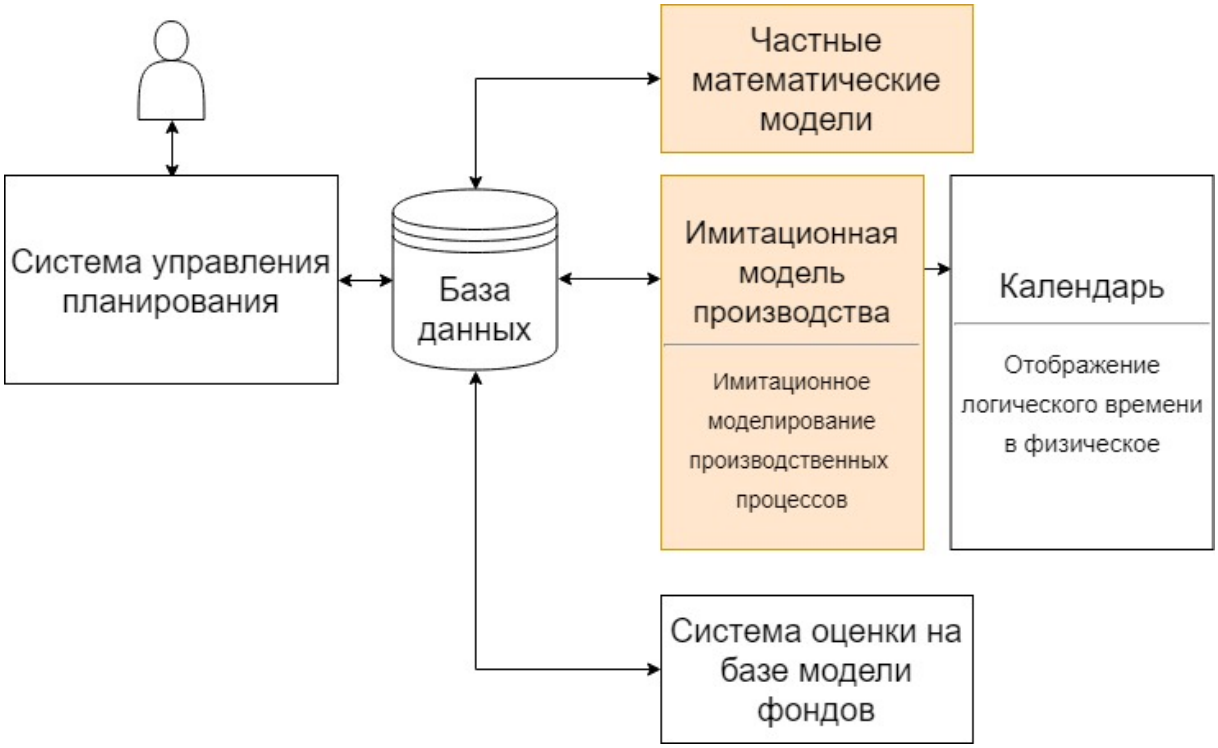

Рисунок 2.1 — Архитектура ПО

На рисунке 2.1 представлена архитектуры программного обеспечения. Данная архитектура содержит следующие элементы:

— пользовательский интерфейс для взаимодействия с системой, который также позволяет получать информацию о работе системы в виде диаграмм, или графиков;

— система управления планирования взаимодействует со всеми элементами системы и является главным распорядителем задач;

— база данных хранит всю информацию о производстве и результаты планирования;

— основная задача имитационной модели заключается в построении оперативного расписания, в котором указаны все операции, время их начала и окончания, время начала и окончания участия производственных ресурсов в выполнения операций;

— календарь обрабатывает логические значения, используемые при планировании, и привязывает их к конкретным датам (физическое время);

— частные оптимизационные модели работают с входными данными имитационной модели. К этим данным применяются алгоритмы оптимизации, зависящие от конкретных целей. Таким целями могут быть: задачи упорядочивания, задачи согласования, задачи распределения, задачи с суммарными критериями оптимизации, задачи с минимаксимальными критериями оптимизации;

— модель оценки фондов работает с производственными мощностями и дают поверхностную оценку осуществимости заданного плана работы на такт (ПРТ), где ПРТ - это оперативный план на один такт сборочного производства, включающего в себя линии и посты, при полной его загрузке - заготовки находятся на каждом посту каждой линии; он обеспечивает максимизацию загрузки ресурсов (опционально - подбор необходимого количества ресурсов)

2.2 Организация системы имитацонного моделирования

2.2.1 Формализация предметной области

Для того, чтобы разрабатывать алгоритмы планирования в первую очередь необходимо формализовать предметную область. В данном случае формализуется сборочный цех со значимыми внутренними особенностями.

Как правило на любом предприятии имеется специальный документ – технологическая карта, детально описывающая весь перечень операций по достижению которых воспроизводится единица продукции. Технологическая карта хранит информацию о зависимостях между операциями, привязках ресурсов к операциям, трудоёмкостях операций, периодичность операций, результат каждой операции.

Далее важно учесть ресурсы предприятия. В рамках сборочного производства такими ресурсами могут быть: персонал, оборудование, организация конвейерной производственной линии.

Последним пунктом для построения модели производства является производственный план. То есть перечень продукции в составе заказа и срок реализации данного заказа. На рисунке 2.2 представлен пример взаимодействия перечисленных выше моделей.

Рисунок 2.2 — Модель производства

2.2.2 Математическое моделирование производственных процессов

Для формализации процесса планирования в работе используются системы неравенств. Пример системы неравенств представлена на рисунке 2.3. Система неравенств состоит из двух частей: статическую и динамическую. Статическая по своей сути описывает последователь-

ность операций, описываемых в технологической карте. Динамическая рассчитывается во время планирования, которое включает ресурсные зависимости операций, поэтому последовательность определяется не только статической частью, но и динамической частью системы неравенств.

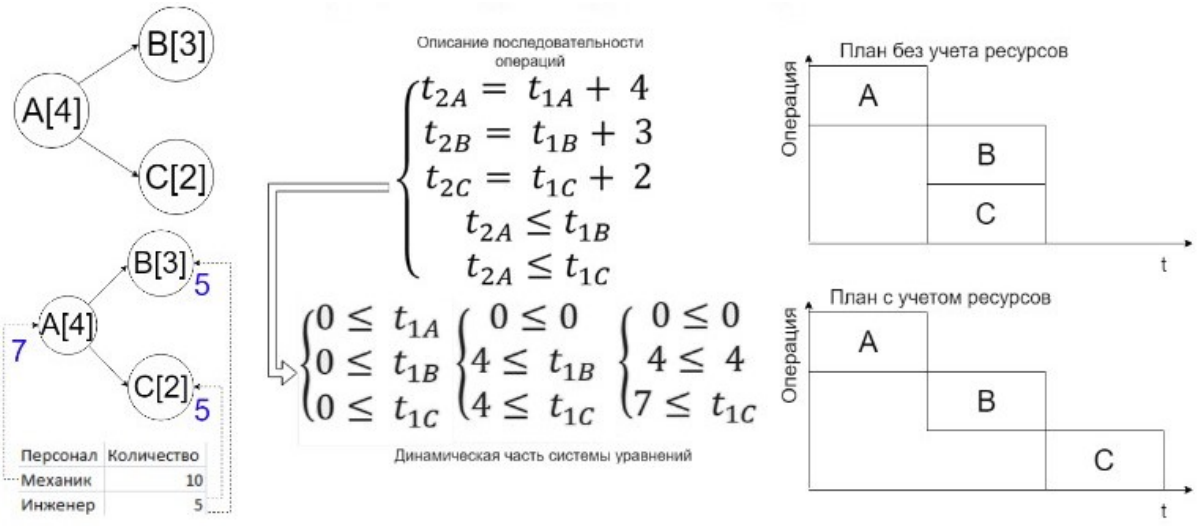

Рисунок 2.3 — Система неравенств

Таким образом планирование делится на шаги, каждый раз при этом формируются новые ограничения вводимые ресурсами.

По результатам анализа работы [13], была составлена градация подходов к имитационному планированию представленная на рисунке 2.4.

Решение задачи планирования аналитическим подходом возможна, при этом основная проблема заключается в сложности реализации всех зависимостей, что делает данный подход сложным в исполнении.

Вторым решением задач имитационного моделирования является алгоритмический подход. Одним из примеров реализации данной идеи является Siemens Plant Simulation.

Система, разрабатываемая в рамках НИР, является гибридной моделью. Данный подход призван перенять лучшее у двух подходов. Там, где задачу можно решить аналитически использоваться аналитический подход, где задача приобретает большое количество зависимостей

задача решается посредством алгоритмов. На рисунке 2.3 приведен пример, того, как выглядит математическая модель производственных процессов, где аналитически выведена статическая часть системы неравенств и посредством алгоритмов генерируется динамическая система неравенств.

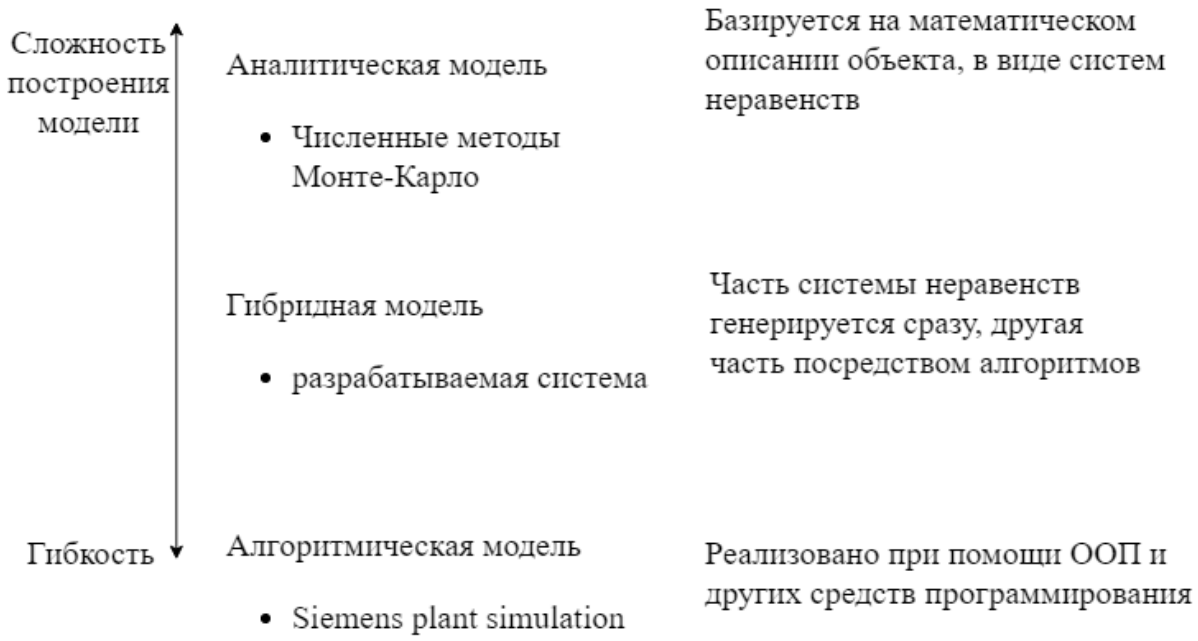

Рисунок 2.4 — Градация подходов к имитационному моделированию

2.3 Частные оптимизационные модели

В данном разделе приведено описание частных оптимизационных моделей. Необходимость применения данных моделей возникла, с появлением задач оптимизации на специфичных производствах. Данная специфика характеризуется уникальным для каждого предприятия структуры, где стандартные методы оптимизации будут не эффективны, либо не будут давать результата.

На текущий момент представлена одна частная оптимизационная модель, задача которой оптимизация сборочных производств. Сборочные производства характеризуются сборочными линиями, где на разных этапах установлены рабочие станции. Рабочий процесс на таком производстве выглядит следующим образом, загатовка переме-

щается от станции к станции, при этом на каждой станции ведется своя независимая работа.

Целью оптимиации является равномерная загрузка всех станций на линии, чтобы избежать простоя ресурсов производства. Для достижения данной цели были созданы модели необходимые для описания сборочного производства. Такими моделями являются ресурс рабочих, а также модель конвейера. Также разработаны методы оптимизации описанные в главе 4.

2.4 Выбор технологии реализации

Планированию как вычислительному процессу необходимо большое количество вычислительных ресурсов и их грамотное использование. Это связано с тем фактом, что при планировании перебираются множество возможных вариантов и выбирается только один лучший (в данном случае имеются в виду алгоритмы оптимального поиска). Когда речь идет о возможности распараллеливания вычислительных ресурсов, требуется технология предоставляющая данную возможность. На данный момент практически все языки программирования имеют функцию распараллеливания, но языки, где это реализовано удобно и без изъянов, немного. Для построения ядра моделирования был выбран многопоточный, компилируемый язык программирования - golang [14].

Язык программирпования golang предоставляет следующие возможности:

1) Скорость обучения. Часто бывает, что разработчики имеют разный уровень подготовки, поэтому необходим язык который позволяет уменьшить затраты на изучения синтаксиса и как можно быстрее сосредоточиться на разработке.

2) Производительность. Golang компилируемый язык [14].

3) Эффективность и возможность многопоточности. В Go есть горутины. Горутинами называют функцию, которая выполняется с другими

гортинами в едином адресном пространстве. Основными преимуществами горутин являются: низкое потребление памяти, минимум накладных ресурсов на организацию, а также простота использования.

4) Встроенный сборщик мусора.

Язык программирования golang вобрал преимущества низкоуровневых языков (сразу компилируется в двоичный код), а также высокоуровневых (имеет сборщик мусора для распределения и удаление объектов). Так как главный акцент, при выборе языка программирования, делался на производительность и поддержку многопоточности, golang является лучшим вариантом для создания ядра моделирования производственного плана.

Данный акцент на быстроте и многопоточности был сделан не случайно. Как будет видно далее, в вопросах поиска оптимальных решений данные особенности golang будут очень востребованы.

3 Система имитационного моделирования.

3.1 Архитектура подсистемы имитационного моделирования

В данном разделе будет рассмотрена работа подсистемы имитационного моделирования производственных процессов (2.1). Работа имитационной модели состоит из следующих этапов:

1) Создание шаблона продукта. Шаблон представляет из себя структуру данных, являющуюся технологической картой одной единицы конкретного продукта.

2) Создание ЦПВ. ЦПВ представляет из себя список структур данных, включающим в себя шаблон и необходимое количество продукта.

3) Следующим этапом является развертывание единого плана на основе заказа. Производственный план описывает совокупность операций[1] и состояние ресурсов.

4) Пошаговая расчет плана с учетом ресурсных ограничений, где каждый раз при расчете переменной из системы неравенств, обновляется информации о состоянии ресурсов.

Работа имитационного моделирования представлена на рисунке 3.1

[1]Так как технологическая карта описывается набором операций, а заказ описывается перечнем технологических карт и их количеством, появилась возможность описать план совокупностью операций.

Рисунок 3.1 — Визуализация работы имитационного моделирования

3.2 Входные и выходные данные

3.2.1 Входные данные

В данном разделе приведена структура входных данных для имитационной модели.

— ЦПВ - цеховая последовательность выпуска (выпуск продукции осуществляется в порядке, в котором они записаны в ЦПВ):

а) наименование продукции;

б) количество единиц продукции;

в) технологическая карта.

— Технологические карты, подробнее смотри (4.1.2).

— Производственные ресурсы;

— *Трафик доступности ресурсов*;

— *Структура цеха*;

— *Минимальное время участия ресурса в операции*;

— *Штрафы за переключение ресурса с одной операции на другую.*

3.2.2 Выходные данные

— Расчетные данные:

а) перечень всех операций для всех единиц продукции ЦПВ;

б) время начала всех операций;

в) время окончания всех операций;

г) *производственные ресурсы, привлекаемые к выполнению операции*;

д) *время начала и конца участия ресурса в операции*;

е) *загрузка производственных ресурсов*.

— Визуализация данных пользователю:

а) расписания для производственных ресурсов;

б) диаграммы Ганта выпуска разных видов продукции;

в) визуализация загрузки производственных ресурсов.

3.3 Этапы имитационного моделирования

В данном разделе подробно описаны все этапы имитационного моделирования. На изображении (3.2) приведена диаграмма потоков данных.

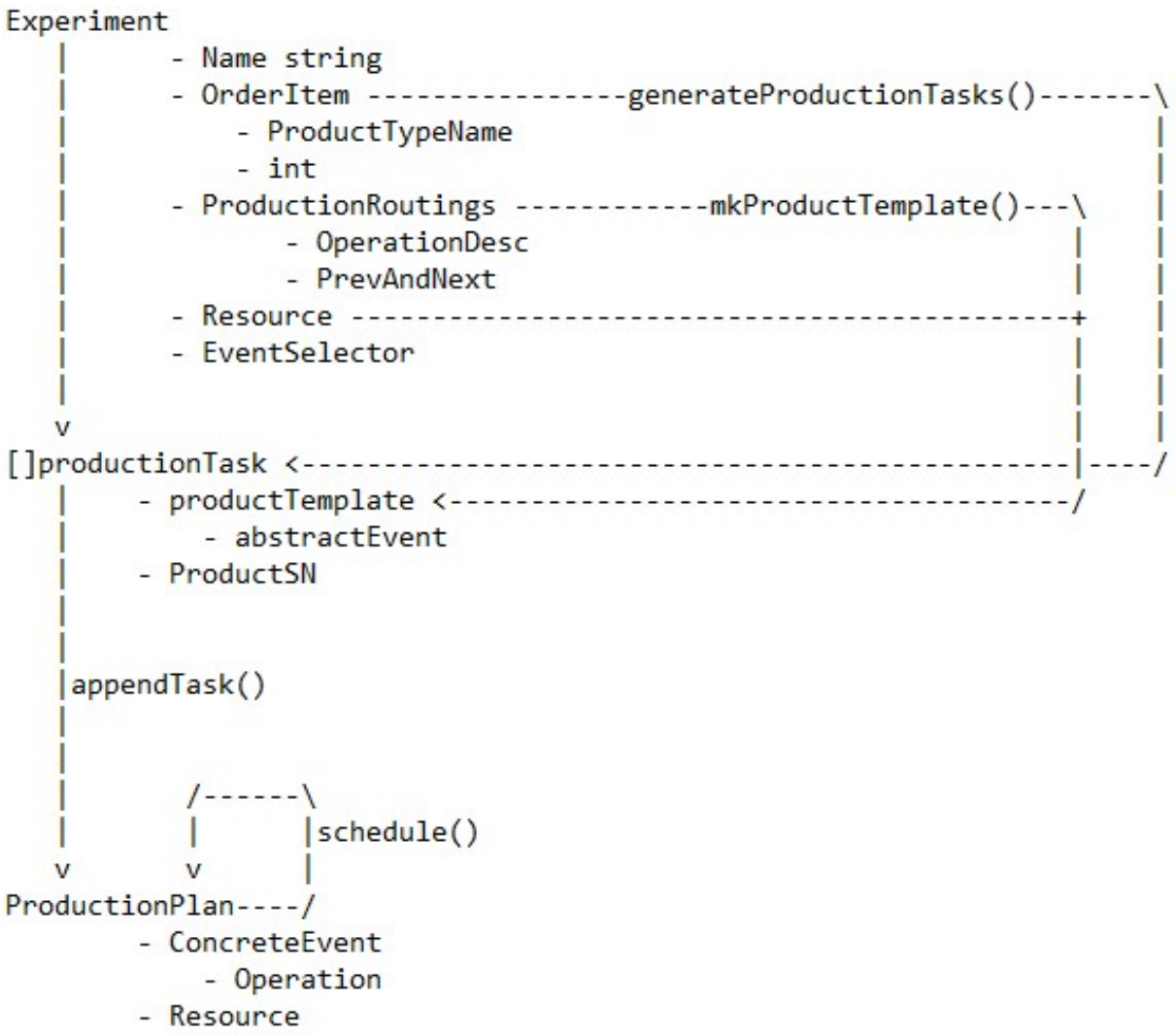

Рисунок 3.2 — Диаграмма потоков данных

3.3.1 Предобработка технологических карт

Входные данные

Входными данными для данного элемента ПО является технологическая карта с конфигурациями ресурсов[1].

Выходные данные

Выходными данными являются технологическая карта с зафиксированными значениями.

[1]Примером конфигурации ресурса может являться вычисление длительности операции на основе количества персонала, привязанного к операции.

Процесс предобработки

Одним из первых этапов работы алгоритма имитационного моделирования является подготовка входных данных, полученных из базы данных.

На текущий момент реализована функционал отвечающий за предварительную обработку. Одной из таких функций является расчет длительности операций. Так как длительность операций является переменной величиной, зависимой от трудоемкости и количества людей выполняющих операцию, существует четыре настройки ресурса:

1) минимальная, при этом на операцию назначается минимальное количество людей, длительность операции становится максимальной;

2) максимальная, при этом на операцию назначается максимальное количество людей, длительность операции становится минимальной;

3) средняя при этом на операцию назначается среднее количество людей, длительность операции становится средней;

4) случайная, при этом количество персонала на операцию выбирается случайным образом.

Также к предварительной настройке относится алгоритм выбора операции[1]. На данный момент существует две конфигурации:

1) стандартная, при которой выбор осуществляется строго по порядку расположения в структурах данных;

2) случайная, при этом случайным образом выбирается одна из возможных операций, которые доступны в данный момент.

Возможность менять входные данные является важным этапом обработки входных данных полученных из базы данных, позволяя при этом подготовить данные для дальнейшей обработки, а также гибко менять параметризуемые величины, чем и достигается большая вариативность необходимая для поиска оптимального значения.

[1]В тех случаях, когда на этапе планирования есть независимые операции

Реализацию функции предобработки технологических карт представлена в Приложении А, Листинг 1.

3.3.2 Создание шаблона продукта

Входные данные

Входными данными для данного элемента имитационной модели является технологическая карта с фиксированными значениями (прошедшая обработку после базы данных).

Выходные данные

Выходными данными является структура данных, включающая в себя набор событий и привязок ресурсов к каждому событию.

Описание реализации

Как говорилось ранее, любая технологическая карта состоит из перечня операций. В процессе работы было принято решение разделить операцию на события, это было вызвано удобством представления аналитических уравнений.

Основной задачей процесса создания шаблона продукта является подготовка необходимых структур данных для дальнейшей работы имитационной модели. На диаграмме потоков данных 3.2, данный процесс именуется как - mkProductTemplate(). В результате работы mkProductTemplate(), создается шаблон продукта необходимый для структуры описывающий заказ.

Реализация функции создания шаблона технологической карты представлена в Приложении А, Листинг 2.

3.3.3 Создание заказа

Входные данные

Входными данными является перечень продуктов и количество каждого продукта из этого перечня.

Выходные данные

Выходными данными является список структур описываемых в разделе 3.3.2, то есть шаблон и количество данного продукта.

Описание реализации

На диаграмме потоков данных 3.2 процесс создания заказа называется - generateProductionTasks(). Основной задачей функции заказа является генерация шаблонов продуктов в зависимости от их количества, при этом каждый однотипный шаблон имеет уникальный идентификатор (серийный номер).

Реализация функции создания заказа представлена в Приложении А, Листинг 3.

3.3.4 Планирование

Входные данные

Входными данными для функции планирования является производственное задание на одну единицу продукции, полученное из списка сформировавшемся на этапе создания заказа (3.3.3).

Выходные данные

Выходными данными является производственный план, состоящий из событий, операций и состояния ресурсов.

Описание реализации

Основной задачей этапа планирования заключается в составлении плана, согласно которому выполняется привязка каждой операции для каждой единицы продукции к временным интервалам, конкретному работнику и конкретным производственным средствам. На диаграмме потоков данных 3.2 процесс отвечающий за создание задания на одну единицу продукции называется appendTask(). Задание включает в себя шаблон технологической карты, название продукта, а также его серийный номер. Процесс формирования заданий работает до тех пор, пока не учтёт всю информацию из полученного заказа. Следующим этапом является процесс планирования на диаграмме потоков данных 3.2 данный процесс называется – schedule(). Именно на данном этапе в работу включается модель ресурсов, описываемая далее в разделе 3.3.5. Основными особенностями данного этапа является сопоставление требованиям ресурсов к операциям, к ресурсным возможностям предприятия, от того строится реальный план, учитывающий ресурсные ограничения.

Результирующая блок-схема подсистемы имитационного моделирования изображена на рисунке 3.3.

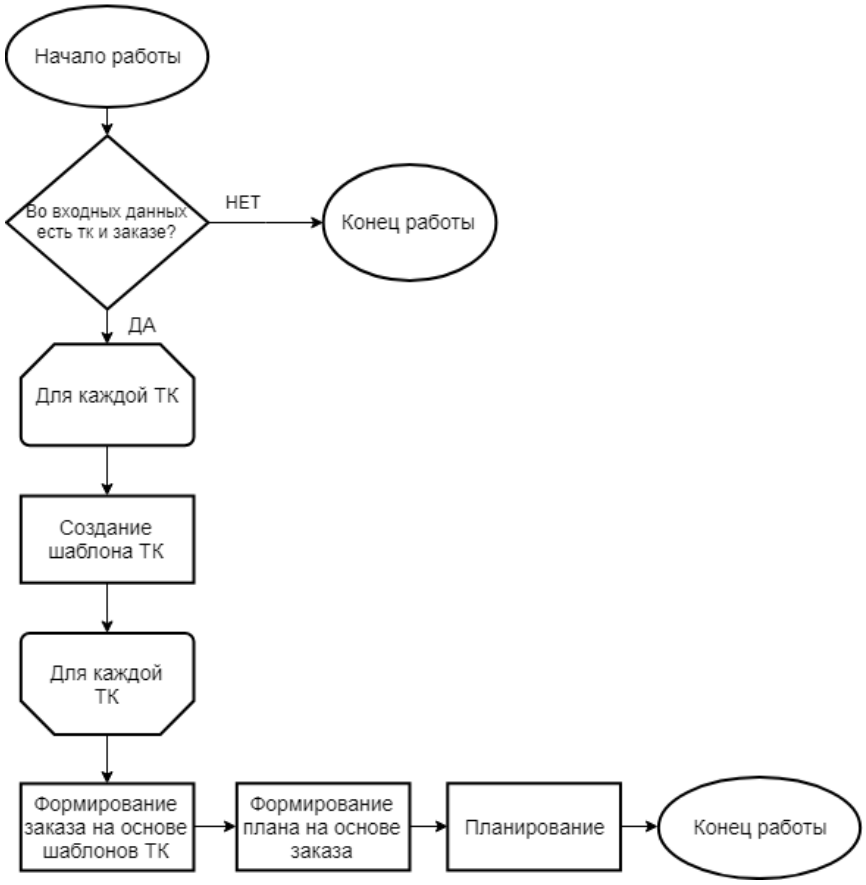

Рисунок 3.3 — Схема работы имитационного моделирования

Реализация алгоритма планирования представлена в Приложении В, Листинг 4,5.

3.3.5 Алгоритм модели ресурсов

В процессе создания оперативного плана, для получения корректной оценки времени выполнения операции или набора операций СПП необходимо ввести систему ограничений, которая будет отражать как ресурс, участвующий в операции может влиять на её время выполнения. Это привело к созданию модели ресурсов накладывающей ограничения на выбор операции для расчета ядром имитационного моделирования. Под ресурсом подразумевается любое устройство,деталь,инструмент или средство,за исключением сырьевого материала и промежуточного продукта, находящиеся в распоряжении предприятия для производства товаров и услуг. В соответствии с данным определением к ресурсам относятся в том числе и человеческие ресурсы, которые в данной

системе не рассматриваются с точки зрения поведения или других аспектов человеческой жизни, а лишь с точки зрения возможности выполнить конкретную задачу. Также необходимо обозначить, что в данном разделе под моделью ресурса будет пониматься упрощенная модель реального ресурса, отражающая его основные (в рамках выполняемых операций) характеристики.

Каждая модель ресурса представляет из себя структуру данных, которая должна реализовывать три метода (Листинг 3.1):

— метод привязки операции к модели ресурса;

— метод, осуществляющий проверку возможности выполнения данной операции моделью ресурса;

— метод, осуществляющий логику работы и в котором происходит изменение состояния данной модели.

Листинг 3.1 — Описание методов ресурса

```
1  type Resource interface {
2      Bind(conf interface{}, events ...*ConcreteEvent)
3      Constrain(event *ConcreteEvent) (int64, bool)
4      Done(event *ConcreteEvent)
5      Clone() Resource
6  }
```

Под привязкой операции к модели подразумевается добавление операции в очередь на выполнение и, если это первая привязанная для данного продукта операция, то добавление продукта в очередь на распределение. Привязка осуществляется в начале работы системы, что позволяет ресурсам манипулировать ядром имитационного моделирования разрешая или запрещая выбирать привязанные к ним операции для расчета, что может повлечь за собой изменение последовательности выполнения операций и, соответственно, расчетного времени выполнения карты технологического процесса.

Проверка производится во время работы системы и именно здесь происходит отбор операций в соответствии с внутренним состоянием модели.

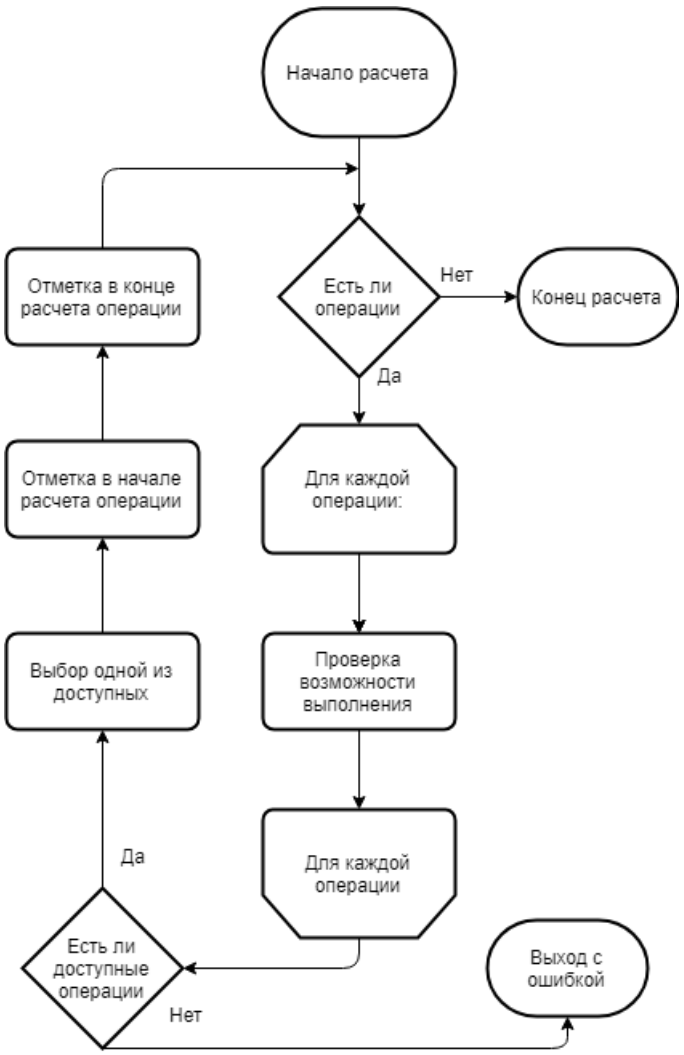

Рисунок 3.4 — Схема работы ядра моделирования с ресурсами в процессе расчета оперативного плана

Логика осуществляется при выборке операции ядром и для каждой вызывается два раза: чтобы отметить состояние модели в начале и в конце расчета операции (см. 3.4).

Одной из реализованных моделей ресурсов является модель рабочих. В предыдущем разделе уже было сказано, что ресурс рабочих рассматривается с точки зрения необходимого элемента функционирования предприятия, при этом упускаются особенности жизнедеятельности рабочих. Это ведет к тому, что на текущий момент не введены соответствующие погрешности регулирующие различные ситуации[1], что соответствующим образом уменьшает точность модели.

[1]Состояние работника, внеплановые перерывы и т.д.

В данной работе ресурс работника характеризуется общим количеством работников данной профессии (квалификации). Также в виде структуры данных реализовано состояние ресурса, которое включает следующие элементы:

— операции, которые предстоит выполнить рабочим;

— операции, которые уже выполнены;

— количество людей данной профессии на конкретную операцию, конкретного продукта, включая серийный номер;

— привязку каждого конкретного работника к выполняемой операции;

— последний временная метка, на которой остановились работы;

— карта занятости рабочих, смотри рисунок 3.5.

Карта занятости является основным источником информации для системы распределения ресурсов. Как видно из данной карты, первые три операции выполнялись строго последовательно, четвертая независимая от предыдущих была определена системой в самое начало последовательности операций. Таким образом первая и четвертая операция будут выполнены одновременно. Решение о том переносе четвертой операции было принято исходя из следующих факторов. Во первый данная операция не связанна с предыдущими причинно – следственными связями. Во-вторых, есть необходимое число свободных работников на данном промежутке времени.

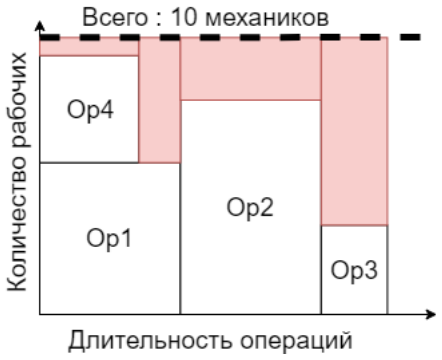

Рисунок 3.5 — Карта занятости рабочих

Может показаться, что информация, описывающая состояние ресурса избыточна, но это не так, потому что не все перечисленные выше элементы участвуют в логике ресурса. Некоторые элементы, такие как привязки каждого конкретного работника к выполняемой операции, используется для расчёта объемно календарного планирования.

3.3.6 Оптимизация на уровне имитационного моделирования

Как упоминалось ранее, в разделе посвященном предобработке технологической карты 3.3.1, исходная ТК предполагает вариативность конфигурации, отсюда следует, что потенциально имитационная модель может сгенерировать большое количество разных оперативных планов.

На рисунке 3.6 приведена зависимость сгенерированного оперативного плана от выбранного критерия оптимальности.

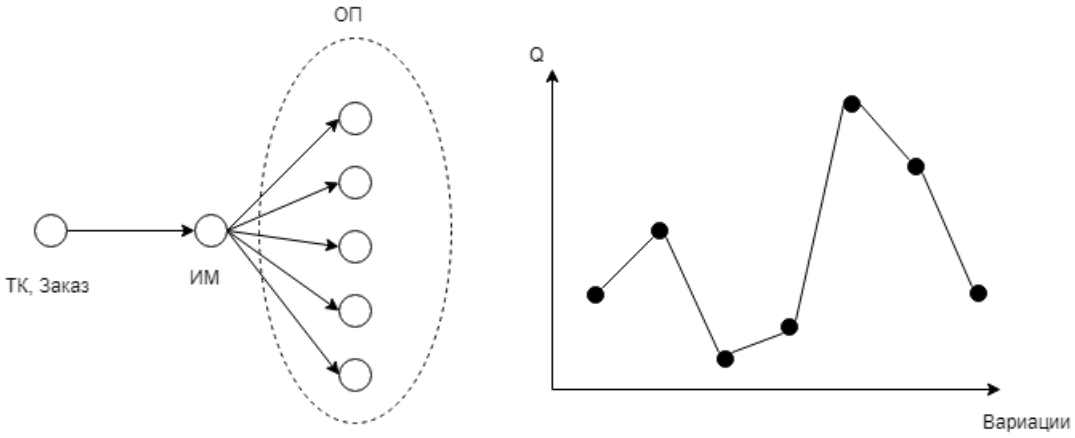

Рисунок 3.6 — Процесс формирования оперативных планов и оценка их качества

В следующем разделе приведены методы, которые позволяют подсистеме оптимизации (рис.3.7) на уровне имитационной модели, найти оптимальную последовательность операций. Критерием оптимальности в данном случае является минимальное время работы линии.

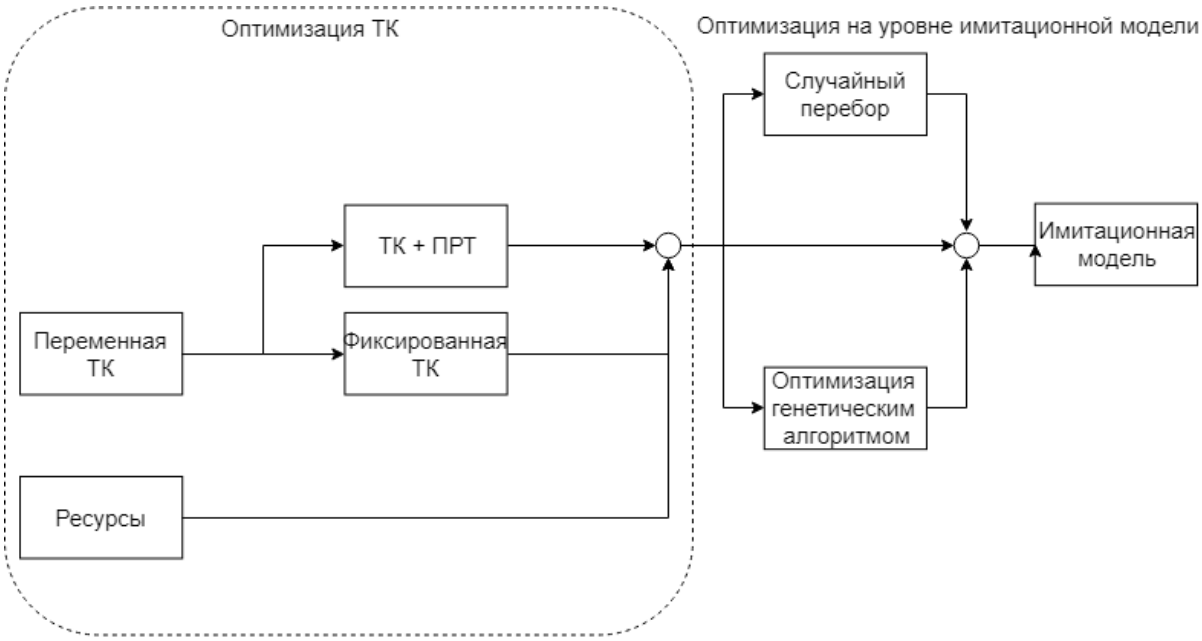

Рисунок 3.7 — Подсистема оптимизации

Метод случайного перебора

Оптимизация на уровне имитационной модели работает непосредственно в процессе планирования, при этом основная задача оптимизации сводится к выбору наиболее «правильной» операции, которая будет выбрана следующей в формировании последовательности операций. Таким образом можно говорить о построении оптимальный последовательности операций, которая учитывает равномерную загрузку ресурсов, а также минимизацию времени работы над заготовкой. На рисунке 3.8 изображены технологическая карта продукта и два плана, который были построены в результате случайного выбора операций.

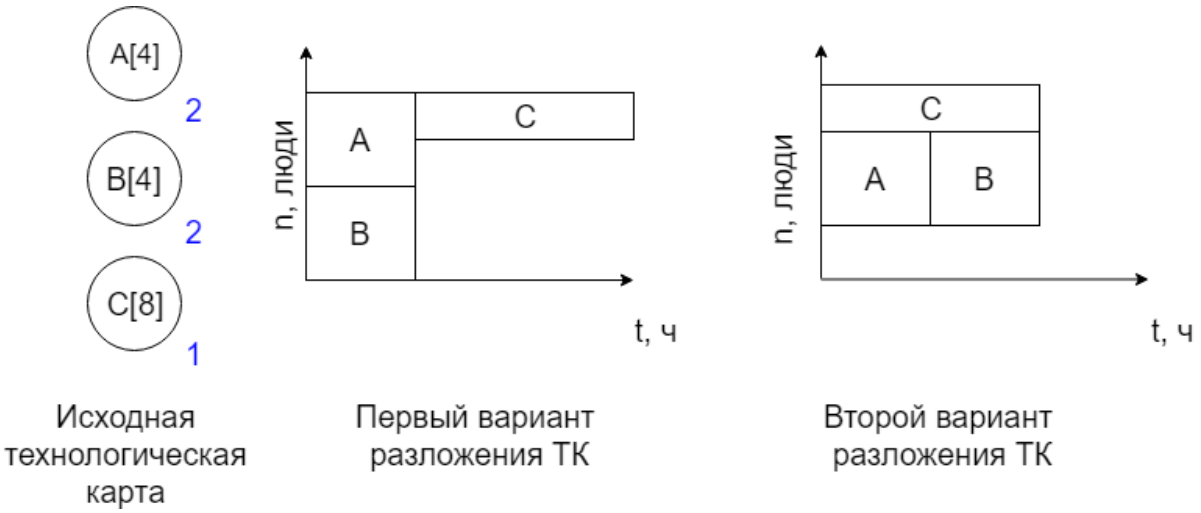

Рисунок 3.8 — Два плана, полученные путем перебора исходной технологической карты

Пример на рисунке 3.8 демонстрирует, то что при комбинировании операций достигается разный результат. При этом стоит отметить, что от выбранного критерия оптимальности, зависит и конечный результат. Так можно заметить корреляцию между равномерной загрузкой рабочих ресурсов и длительностью выполнения работ, которая отражается в том, что равномерное распределение задач среди рабочих ведет и к уменьшению общей длительности работы.

Реализация случайного перебора на уровне имитационной модели приведен в Приложении В, Листинг 6.

Генетические алгоритмы

Основной задачей данного подхода является поиск оптимального значения путем смешивания и введения небольших правок (мутаций) в выборку наилучших решений. Идеей такого рода алгоритмов является направленность поиска, которая позволяет преодолеть плато решений, при которых не происходит изменений в лучшую сторону.

Эксперименты эффективности оптимизации

Результаты экспериментов показали, что эффективность оптимизации во многом зависит от размера и состава входных данных. На рисунке 3.9 продемонстрирован результат работы случайного перебора на основе имитационной модели.

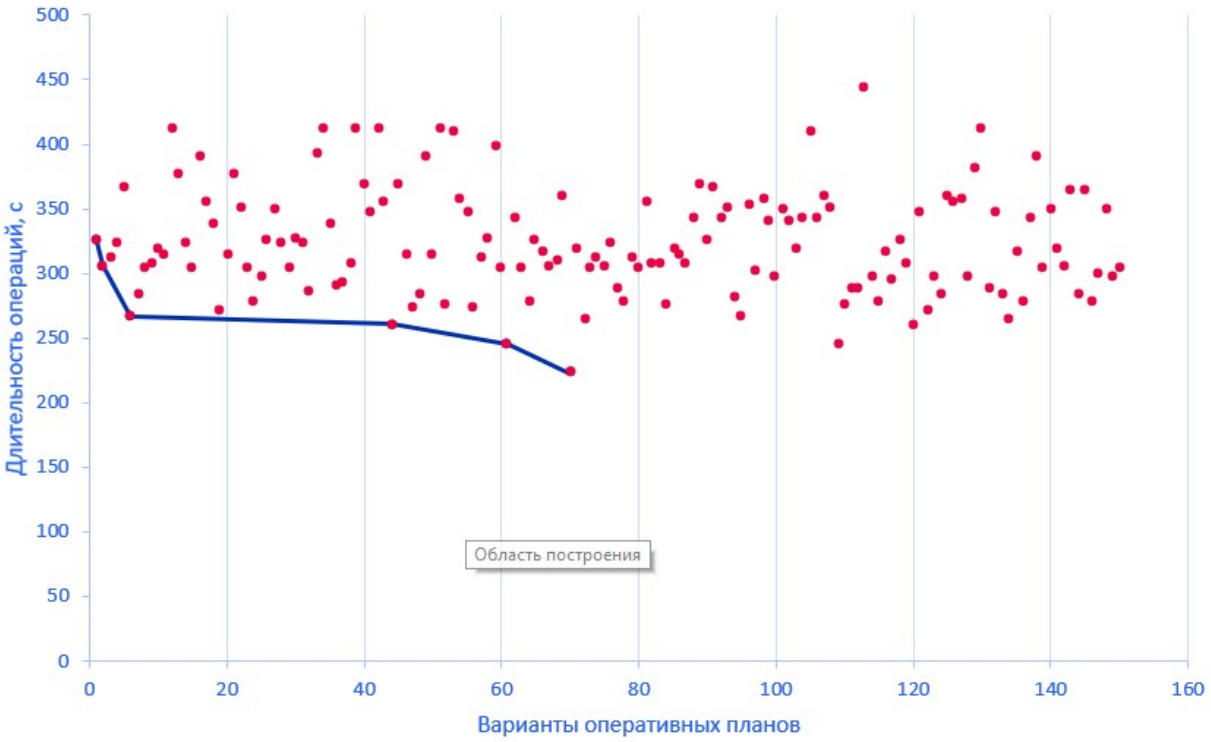

Рисунок 3.9 — Случайный перебор на уровне имитационной модели

Так на входных данных полученных от реального производства, оптимизация случайным перебором показала себя не эффективно. В первую очередь это связано с размером входных данных[1]. В вторых это связано со сложностью алгоритма планирования, так как сам случайный перебор является его частью, что делает его зависимым от реализации имитационной модели.

Также стоит отметить проблемы, связанные с входными данными. Данные проблемы заключаются в вариативности данных. На уровне имитационной модели метод случайного перебора может оперировать только выбором следующей операции на обработку. Таким образом,

[1] 325 операций в одной технологической карте.

если операции на выбор только одна, в случае когда последовательность операций в технологической карте задана строгим образом и такая связь задана для большинства операций, то количество вариантов стремится к минимуму, а значит и целесообразность применения метода ставится под вопросом.

В результате эксперимента была сделаны следующие выводы. Метод случайного перебора зависит от размера входных данных, при этом важно оценивать входные данные на предмет вариативности. В том числе стоит отметить, что сложность задачи определяется пользователем на этапе конфигурации перебора[1], что еще сильнее повышает ценность предварительной оценки технологической карты, с целью оценки надобности алгоритма перебора.

3.4 Результаты работы имитационной модели

Проведение экспериментов над имитационной моделью показали следующие результаты.

Во первых при большом объеме входных данных имитационная модель не растет экспоненциально. Данный факт подтверждается проведенными экспериментами. Соответствующий график приведен на рисунке 3.10. Более того, из данного графика видно, что рост сложности приближен к линейному.

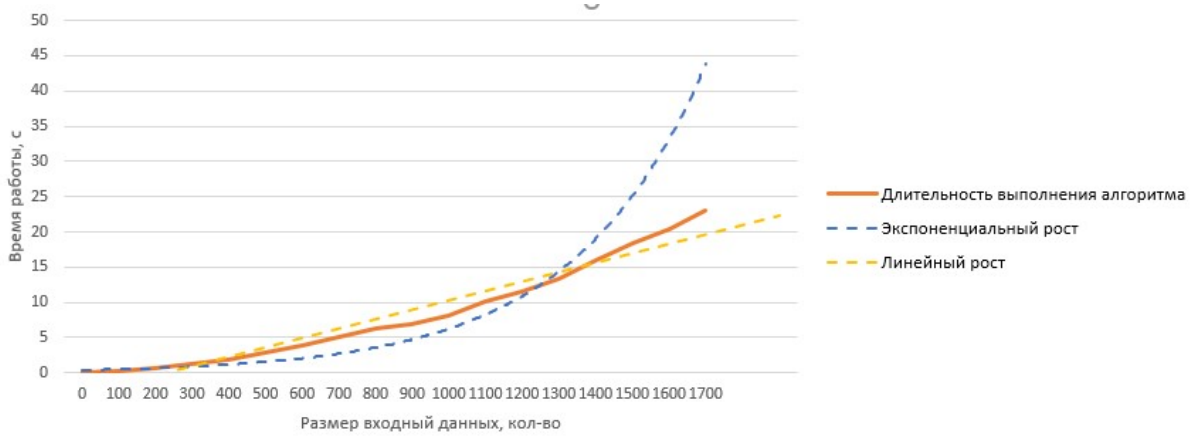

Рисунок 3.10 — Временная сложность имитационной модели

[1]Имеется ввиду количество созданных планов со случайной последовательностью операций.

Далее приведен график временной сложности расчета оперативного плана для конвейеризированного производства. Данный график приведен на рисунке 3.11. В отличии от предыдущего случая временная сложность данной модели предполагает степенной рост, также стоит отметить большее время расчета по сравнению с обычным планированием при одинаковом размере входных данных.

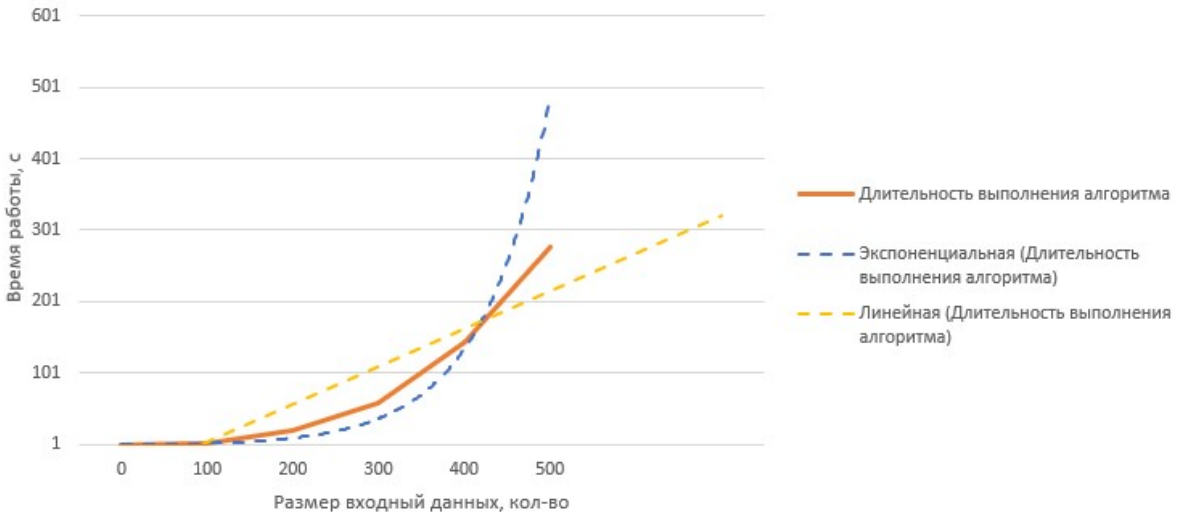

Рисунок 3.11 — Временная сложность имитационной модели конвейеризированного производства

Также была исследована разница между параллельным и последовательным расчетом имитационной модели. Были получены следующие результаты, представленные на рисунке 3.12. Данный эксперимент был проведен при 100, 500 и 1000 запусков планирования. Таким образом видно насколько эффективно справляется со своими задачами параллелизм реализованный в Golang.

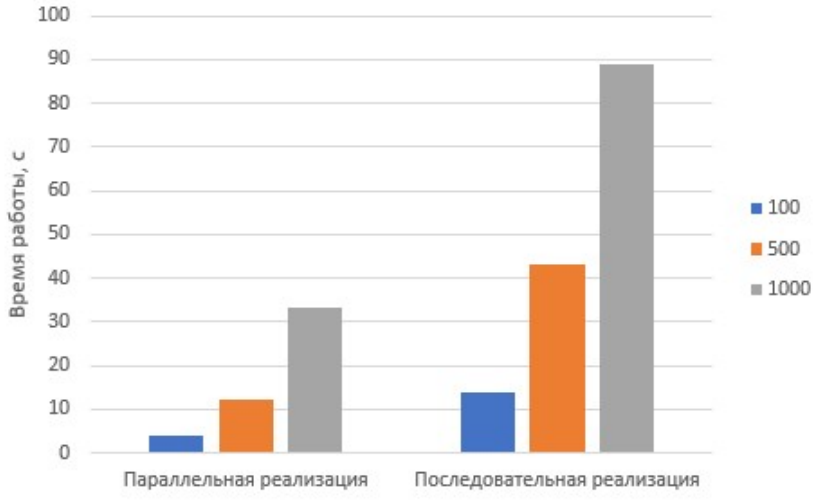

Рисунок 3.12 — Параллельный и последовательный расчет имитационной модели

Тестирование оптимизации случайным перебором приведено в приложении С, Листинг 7.

Во вторых выяснилось, что потенциально основной проблемой в будущем может оказаться сбор реальных данных для работы имитационной модели. В рамках дипломной работы проверка имитационной модели проверялась на реальных технологических картах, при этом возник закономерный вопрос о сложности получения и организации данных о предприятии.

4 Балансировка линии сборочных производств

4.1 Назначение и условия применения алгоритмов плана работы на такт

4.1.1 Назначение плана работы на такт

На рисунке 3.7 приведена подсистема оптимизации. В предыдущем разделе были рассмотрены методы оптимизации на уровне имитационной модели, в данном разделе будут рассмотрены методы оптимизации на уровне входных данных, то есть предобработка технологической карты.

В связи с тем, что одной из основных целей проделанной работы является оптимизация производственного процесса, в рамках имитационной модели были реализованы оптимизационные задачи. Но как было замечено ранее, подходы к оптимизации не всегда эффективно справлялись со своей задачей, ввиду примитивной, но рабочей реализации (ввиду специфичности данных, с которыми работает имитационная модель). Основная проблема заключалось в сложности исходной задачи, где время работы алгоритмов случайного перебора определяется факториальным временем. Таким образом необходимо знать размер входных данных, и исходя из этого принимать решение о целесообразности применения полного перебора.

Для того, чтобы уменьшить влияние входных данных были разработаны разные вариации алгоритмов составления плана работы на такт. Назначением плана работы на такт является обеспечение максимизации загрузки ресурсов, что приведет к повышению экономической эффективности производства. ПРТ представляет из себя оперативный план на один такт сборочного производства, включающего в себя линии и посты, при полной его загрузке - заготовки находятся на каждом посту каждой линии.

4.1.2 Входные данные

В данном разделе приведена структура данных для алгоритма работы оптимизации плана работы на такт:

1) Технологическая карта включает:

   — операции и их причинно-следственные связи;

   — рабочие места на заготовке, макс. кол-во работников на рабочем месте;

   — задействованные ресурсы для каждой операции:

      а) работники соответствующих профессий (min, max, трудоёмкость, диапазон ожидаемого отклонения трудоёмкости[1]);

      б) рабочее место на заготовке;

      в) *номер поста для операции*;

2) Параметры производства включают:

   — организация производства:

      а) кол-во линий;

      б) *кол-во постов*;

      в) доступность/недоступность рабочих мест на заготовках на постах[2];

      г) *такт производства*;

   — персонал:

      а) *макс. кол-во работников в цеху*;

      б) *кол-во работников каждой профессии*;

[1] Позволяет учитывать при планировании ожидаемые отклонения по оптимистическому/пессимистическому/случайному сценарию.

[2] К примеру, работы на крыше являются высотными и требуют специальных ограждений.

в) *привязка работников к посту/линии*;

г) сменный график по профессиям;

3) Настройки расчёта (*опционально*):

— изменение числа работников в процессе работы над операцией:

а) возможность изменение числа исполнителей операции;

б) минимальное время участия – не имеет смысла привлекать работника на очень короткий промежуток времени, так как больше на включение в операцию потратит.

в) временные потери на смену рабочего места, включение в выполняющуюся операцию;

— выравнивание сменного графика, производственного такта и операций (по смене, по 1/2, по 1/4, ...);

— синхронизация работы линий:

а) задается максимально допустимое время отклонения сборки на одном посту между линиями.

б) задается допустимый временной сдвиг между линиями.

4) Оценка качества варианта:

— настройка отклонения длительности работы от производственного такта (-0.1, +0.01);

— настройки максимальной / минимальной загрузки персонала (0 - особый случай);

4.1.3 Выходные данные

Выходными данными является ПРТ, что включает:

1) время начала и конца операции (длительность);

2) кол-во работников на операцию;

3) расписание занятости ресурсов во времени, загрузка;

4) временные потери на перемещение персонала (в случае изменения числа работников в процессе работы над операцией).

4.2 Балансировка сборочной линии

Производственная сборочная линия была впервые представлена Генри Фордом в начале 1900-х годов. Она была разработана с целью повышения эффективности, производительности изготовления конкретного продукта. Базовая сборочная линия состоит из нескольких рабочих станций, расположенных последовательно, где каждая станция соединена погрузочно-разгрузочным устройством. Движение материала или заготовки по сборочной линии начинается всегда с первой станции, оно осуществляется с заданной скоростью – она определяется тактом производственной линии. Станцией может считаться любая точка конвейера, на которой выполняется обработка заготовки - машинами, роботами и или людьми. Как только заготовка поступает на станцию начинается выполнение соответствующей операции (определяемой технологической картой), после окончания работ она подается на следующую станцию. Время, необходимое для выполнения операции на каждой станции, называется временем процесса [15]. Цикл производственной линии представляет собой общее время прохождения заготовкой всех станций конвейера с учетом простоев между операциями. Длительность цикла сборочной линии определяется желаемой производительностью. Этот уровень производства устанавливается таким образом, чтобы желаемое количество конечного продукта было произведено в течение определенного периода времени [16]. Для того чтобы сборочная линия поддерживала определенную производительность, такт производственной лини (время самой длительной операции конвейера) не должен превышать среднюю длительность процессов на станциях. Если время обработки на некоторой станции превышает среднюю длительность операции конвейера, говорят, что на этой станции присутствует простой. Одним из основных вопросов, касающихся

организации сборочной линии, является порядок выполнения задач. Проблема балансировки сборочной линии (ALBP) возникла вскоре после широкого распространения сборочных линий в промышленности. Хельгесон и др. [17] были первыми, кто предложил рассмотрение ALBP, как проблемы, требующей исследований, тогда как Сальвесон [18] был первым, кто опубликовал предложил математическую формализацию проблемы. Однако в течение первых сорока лет существования сборочной линии для балансировки линий использовались только методы проб и ошибок. С тех пор было разработано множество методов для решения различных форм ALBP. Сальвесон [18] сделал первую математическую попытку, решив задачу в виде линейной программы. Гутьяр и Немхаузер [19] показали, что проблема ALBP относится к классу NP-сложных задач комбинаторной оптимизации. Это означает, что оптимальное решение не гарантируется для задач значительных размеров. Поэтому эвристические методы стали наиболее популярными методами решения проблемы.

Задачи балансировки линии:

— уменьшение количества рабочих станций при заданном цикле;

— уменьшение цикла при заданном количестве рабочих станций;

— уменьшение общего времени простоя;

— уменьшение общего объекта или длины линии.

Классификация проблемы ALB основана главным образом на целевых функциях и структуре сборочной линии. Различные версии проблем ALB представлены на рисунке 4.1 [20].

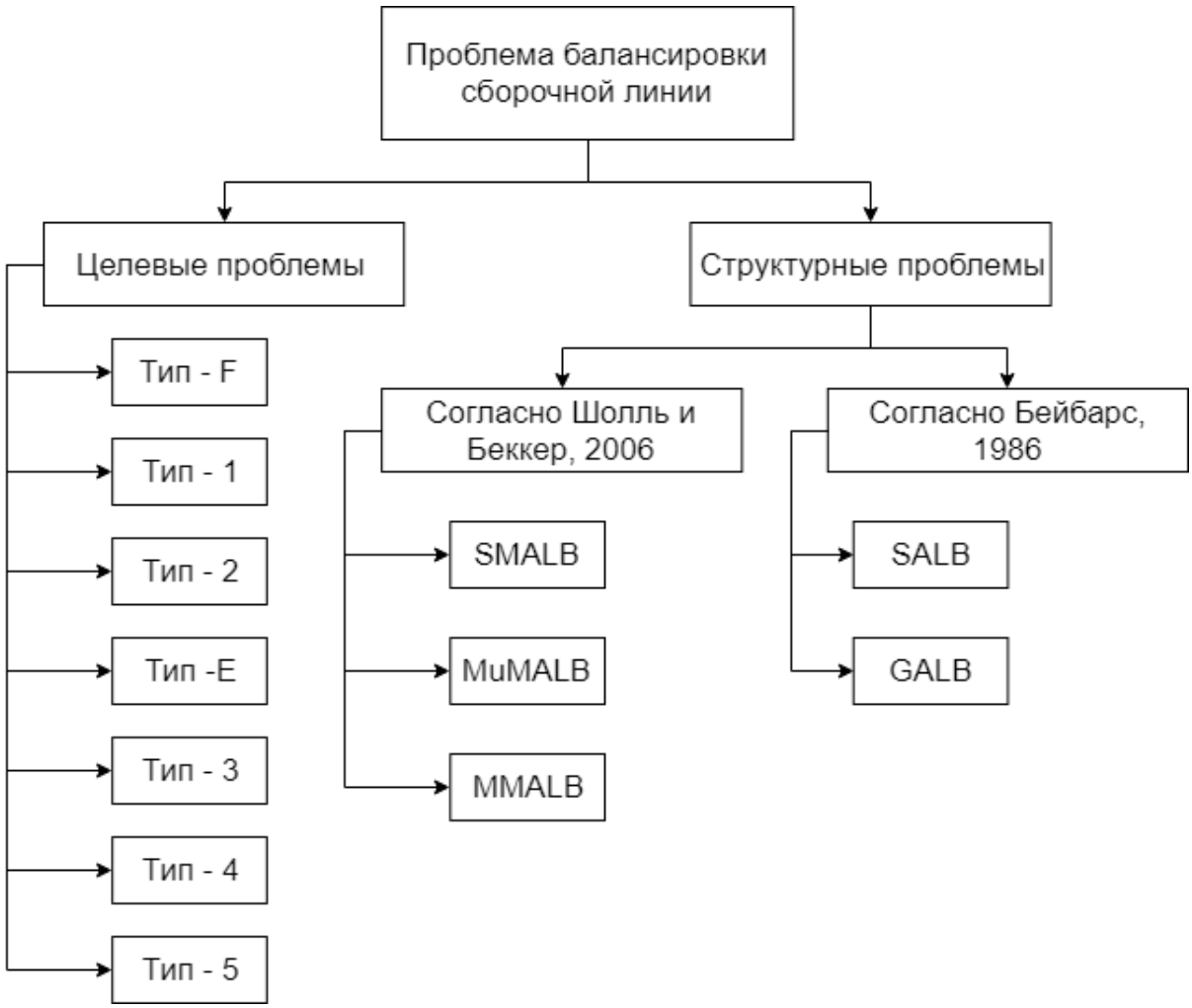

Рисунок 4.1 — Классификация ALBP

4.2.1 Проблемы базирующиеся на целевых функциях

В данном подразделе рассматриваются целевые задачи балансировки линии. Далее перечисленные все обозримые проблемы приведенные на классификации 4.1 и представленно их краткое описание.

— Тип F: Рассматривает возможность создание линии при заданном количестве рабочих станций и цикле.

— Тип 1: Рассматривает задачу уменьшения количества рабочих станций, при фиксированном времени цикла.

— Тип 2: Рассматривает задачу уменьшения времени цикла, при фиксированном количестве рабочих станций.

— Тип E: Данный тип является самой общей версией задачи расчета ПРТ и рассматривает получение максимальной эффективности линии при минимальном цикле и количестве станций.

— Тип 3: Рассматривает задачу увеличение плавности рабочей нагрузки.

— Тип 4: Рассматривают увеличение синхронности работы, используется в тех случаях, когда нужно производить быстро однотипный продукт.

— Тип 5: Рассматривает типы 3 и 4 для нескольких продуктов.

В рамках данной работы рассматривается функция повышения эффективности загрузки. Эффективность сборочной линии подразумевает равномерную загрузку всех рабочих станций на сборочной линии.

4.2.2 Проблемы основывающиеся на структуре линии

В данном разделе рассматриваются структурные задачи балансировки линии. Далее перечислены структурные проблемы приведенные на рисунке 4.1 и их описание [20].

— SMALB: Данная проблема затрагивает структуру, когда на линии производится один тип продукта.

— MuMALBP: Затрагивает проблемы производства более одного типа продукта партиями на одной линии.

— MMALBP: Затрагивает производство разных типов продуктов на одной линии в любом порядке, без времени переключения (имеется ввиду время необходимое для переоснастки рабочих мест на линии для производства нового типа продукта).

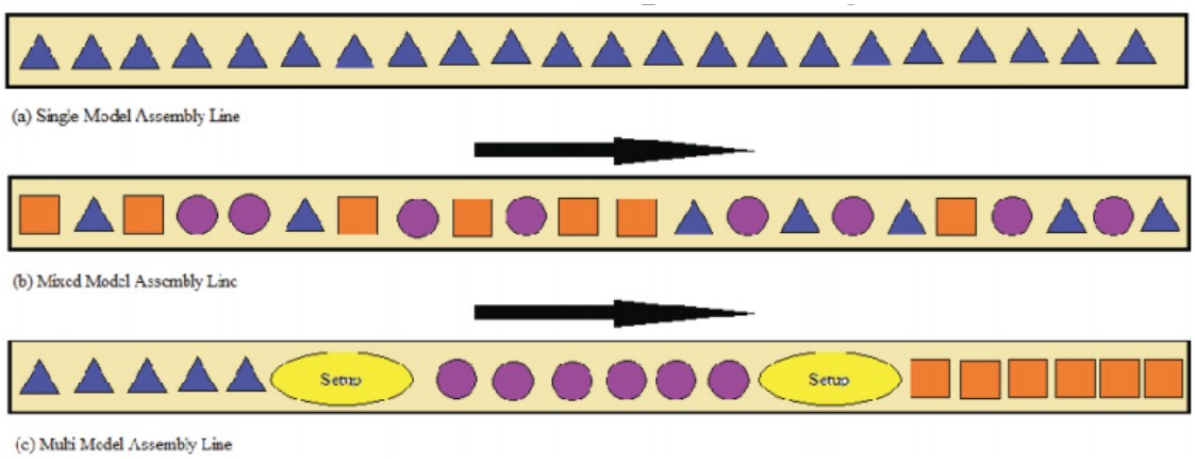

Рисунок 4.2 — Структура линий из классификации, изображенной на рисунке 4.1

Созданная имитационная модель позволяет игнорировать структурные особенности линии. Основные ограничения, которые невозможно игнорировать включены в технологическую карту.

4.3 Обзоры методов решения проблем балансировки линии

На текущий момент известны множество подходов для решения проблемы балансировки линии. Наиболее популярные подходы и методы приведены на рисунке (4.3).

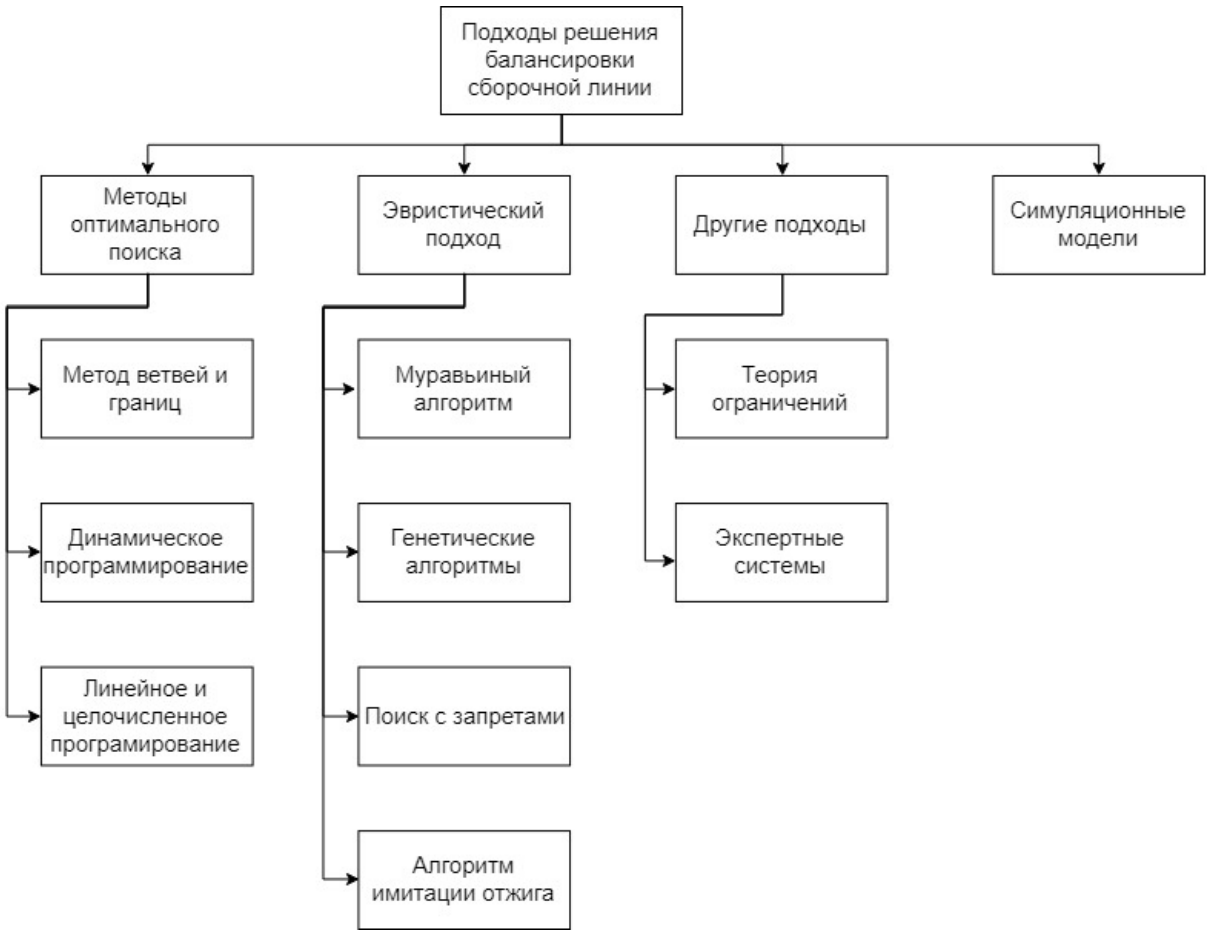

Рисунок 4.3 — Различные процедуры решения для ALBP

4.3.1 Методы оптимального поиска

Методы оптимального поиска основаны на математических подходах и позволяют найти из множества объектов оптимальный, который соответствует заданным критериям. Одной из основных проблем данного подхода является вычислительная сложность, что в контексте поиска наилучшей конфигурации конвейера, приводит к существенному ограничению использования методов оптимального поиска. В следующем подразделе будет рассмотрен метод оптимизации случайного перебора, который может существенно снизить вычислительную сложность алгоритма.

Метод ветвей и границ

В общем случае метод ветвей и границ позволяет отсеять подмножество допустимых решений, заведомо не содержащих оптимальных решений. Поиск оптимизируемого подмножества сводиться к поиску станции на которой время выполнения всех операций является максимальным из возможных.

В контексте балансировки линии задача будет сформулирована следующим образом:

1) Из всех возможных станций выбор наиболее загруженной с целью полного перебора всех возможных вариантов последовательностей и поиск оптимального решения. В результате может быть два возможных варианта, либо перетасовка операций действительно позволила использовать неиспользуемые ресурсы, либо перетасовка ни к чему не привела. На рисунке 3.8 продемонстрированы два варианта разложения технологических карт.

В первом варианте разложения технологической карты две независимые операции A и B выполняются параллельно при этом задействованы все ресурсы в промежутке от нуля до четырех часов. Далее выполняется операция C, которая требует для выполнения одного работника, при этом трое рабочих остаются без работы на протяжении восьми часов.

Во втором варианте операция C выполняется параллельно с операциями A и B при этом общее выполнения всех операций сократилось до восьми часов и на протяжении 8 часов один незадействованный рабочий.

Данный пример демонстрирует один из возможных способов оптимизации путем перетасовки операций, опирающийся на причинно-следственную связь операций. Где из всех возможных множеств, выбирается только одно подмножество, которое с большой вероятностью содержит приближенный к оптимальному результат.

2) В тех случаях, когда оптимизация перетасовки операций не принесла результатов, используется подход основанный на изменении привязок ресурсов к операциям. Так как длительность операции не задается, а рассчитывается на основе входных данных, изменение этих данных позволяет гибко менять длительности операций, и таким образом влиять на загрузку ресурсов.

4.3.2 Эврестический подход

Методы эвристики позволяют нивилировать проблемы комбинаторной сложности задачи, но при этом результат не всегда будет являться оптимальным. Также эффективность работы алгоритмов эвристики во многом зависят от подхода. На рисунке 4.3 приведены пять различных эвристических алгоритмов, которые использовались для решения проблемы балансировки линии. Одним из наиболее эффективных показал себя генетический алгоритм поиска. Рассмотрим подробно, как работает генетический алгоритм в контексте балансировки линии.

Генетический алгоритм

Генетические алгоритмы хорошо подходят для решения задач планирования производства, потому что в отличие от эвристических методов генетические алгоритмы работают на совокупности решений, а не на одном решении. В производственном планировании эта совокупность решений состоит из множества ответов, которые могут иметь разные, иногда противоречивые цели. Например, в одном решении оптимизировать производственный процесс, который будет завершен за минимальное время. В другом решении оптимизировать для минимального количества дефектов.

По мере того как увеличиваются количество целей, которые пытаемся достичь, также увеличивается количество ограничений на проблему и аналогичным образом увеличиваем сложность. Генетиче-

ские алгоритмы идеальны для задач такого типа, когда пространство поиска велико, а количество возможных решений мало.

Чтобы применить генетический алгоритм к задаче планирования, необходимо сначала представить каким образом обозначить геном. Одним из способов представления генома планирования является определение последовательности задач и времени начала этих задач относительно друг друга. Каждое задание и соответствующее время его запуска представляют собой ген.

Определенная последовательность задач и времени начала (гены) представляет один ген в нашей популяции. Чтобы убедиться, что геном является возможным решением, надо чтобы он соответствовал ограничениям приоритета. Далее генерируется начальная популяция, используя случайные времена начала в пределах ограничений предшествования. С помощью генетических алгоритмов берется начальная популяция и скрещивается, комбинируя гены с небольшим количеством случайности (мутации). Потомки этой комбинации выбираются на основе функции приспособленности [1], которая включает одно или много наших ограничений, таких как минимизация времени и минимизация дефектов. Данный процесс продолжаться либо в течение заранее выделенного времени, либо до тех пор, пока не найдется решение, которое соответствует минимальным критериям. В целом каждое последующее поколение будет иметь более высокую среднюю пригодность, то есть займет меньше времени с более высоким качеством, чем предыдущие поколения. Также может потребоваться добавить дополнительные значения пригодности, такие как минимизация затрат; однако каждое добавляемое ограничение значительно увеличивает пространство поиска и уменьшает количество подходящих решений.

[1] Функция приспособленности — вещественная или целочисленная функция одной или нескольких переменных, подлежащая оптимизации в результате работы генетического алгоритма, направляет эволюцию в сторону оптимального решения. Является одним из частных случаев целевой функции

4.3.3 Результаты экспериментов оптимизации плана работы на такт

Оптимизация плана работы на такт, в отличии от оптимизации на уровне имитационной модели, работает над обработкой исходной технологической картой. Данное отношения подходов можно увидеть на рисунке 3.7, демонстрирующей подсистему оптимизации.

Результаты работы алгоритмов ПРТ, как и в разделе 3.3.6, были протестированы на реальных данных. Полученные результаты ничего не показали. Это связано с тем, что в привязках к операциям полностью отсутствовала вариативность, число рабочих на операции было строго фиксировано. Таким образом протестировать эффективность алгоритмов ПРТ на реальных данных оказалось невозможным, поэтому для тестирования была создана искусственная ТК.

Основной целью алгоритмов ПРТ была оптимизация конвейеризированных производств. Для этого были разработаны и протестированы два подхода. Первый подход заключался в оптимальном поиске случайным перебором. Методом оптимизации случайного перебора был выбран метод ветвей и границ, который значительно уменьшал размер входных данных. На тестовой ТК алгоритм оптимального поиска показал себя эффективным ввиду небольшого размера исходной задачи. При этом генетический алгоритм показал подобный результат, при этом была выдвинуто предположение о том, что при большом размере входных данных будет большая разница в эффективности данных подходов в пользу генетических алгоритмов.

Заключение

В рамках данной работы рассматривались задачи построения цифрового двойника производства, имитирующего производственные процессы, а также рассматривались методы поиска оптимального оперативного плана производства.

В рамках работы были достигнуты следующие результаты:

1) создана архитектура подсистемы имитационного моделирования;

2) разработаны алгоритмы генерации оперативного расписания производства.

В ходе работ также были исследованы области оптимизации планирования на производстве и достигнуты следующие результаты:

1) разработана архитектура подсистемы оптимизации;

2) разработаны алгоритмы оптимизации на уровне имитационной модели;

3) разработаны алгоритмы оптимизации плана работы на такт для конвейеризированнных производств;

4) проведено тестирование имитационной модели и функционирования методов оптимизации в различных режимах работы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. С. Гончаров А., М. Саклаков В. Цифровой двойник: обзор существующих решений и перспективы развития технологии // Статья в сборнике трудов Всероссийской научно-практической конференции. 2018. С. 24–26.

2. Олег Новиков. ЧТО ТАКОЕ ИНДУСТРИЯ 4.0? ЦИФРЫ И ФАКТЫ. 2015. URL: http://holzex.ru/chto-takoe-industriya-4-0-tsifryi-i-faktyi.

3. Д.А. Гаврилов. Управление производством на базе стандарта MRP II. Спб Питер, 2003.

4. Ptak C. A., Schragenheim E. ERP: tools, techniques, and applications for integrating the supply chain. St. Lucie Press, 2004.

5. Д. О'Лири. ERP системы. Современное планирование и управление ресурсами предприятия. Выбор, внедрение и эксплуатация. 2004.

6. Разумов И.М., Степанов А.П., Смирнов С.В. Научная организация и нормирование труда в машиностроении. Учебник для вузов. — М.: Машиностроение, 1975.

7. E. Velazco Enio. Simulation of manufacturing systems // International Journal of Continuing Engineering Education and Life-Long Learnin. 1994. 12. Т. 4. С. 80–92.

8. Benedettini Ornella, Tjahjono Benny. Towards an improved tool to facilitate simulation modelling of complex manufacturing systems // International Journal of Advanced Manufacturing Technology. 2008. 07. Т. 43. С. 191–199.

9. G. Holst Lars, Bolmsjö Gunnar. Simulation integration in manufacturing system development: A study of Japanese industry // Industrial Management and Data Systems. 2001. 10. Т. 101. С. 339–356.

10. B Detty Richard, C Yingling Jon. Quantifying benefits of conversion to lean manufacturing with discrete event simulation: A case study // International Journal of Production Research. 2010. 11. T. 38.

11. Robinson Stewart. Simulation: The Practice of Model Development and Use, 2nd edition. 2014. 04.

12. Venkateswaran Jayendran, Son Young-Jun, Jones Al. Hierarchical Production Planning Using a Hybrid System Dynamic-Discrete Event Simulation Architecture. T. 2. 2005. 01. C. 1094–1102 vol.2.

13. Simulation in manufacturing and business: A review / Mohsen Jahangirian, Tillal Eldabi, Aisha Naseer [и др.] // European Journal of Operational Research. 2010. 05. T. 203. C. 1–13.

14. Doxsey Caleb. Introducing Go. 2016.

15. J. SURY R. Aspects of assembly line balancing // THE INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH. 1971. 01. Vol. 9. P. 501–512.

16. Baybars I. A Survey of Exact Algorithms for the Simple Assembly Line Balancing Problem // Management Science. 1986. 08. Vol. 32. P. 909–932.

17. Helgeson W., Birnie D. Assembly Line Balancing Using the Ranked Positional Weighting Technique. // Journal of Industrial Engineering. 1961. Vol. 12. P. 394–398.

18. Salveson M. The assembly line balancing problem // Journal of Industrial Engineering. 1955. 01. Vol. 6. P. 18–25.

19. L. Gutjahr A., Nemhauser G. An Algorithm for the Line Balancing Problem // Management Science. 1964. 11. Vol. 11. P. 308–315.

20. Uddin Mohammad Kamal, Lastra Jose Luis Martinez. Assembly Line Balancing and Sequencing // Assembly Line / под ред. Waldemar Grzechca. Rijeka: IntechOpen, 2011. URL: https://doi.org/10.5772/19953.

Листинг 1 содержит реализацию функции предобработки технологической карты.

Листинг 1 — Предобработка технологической карты

```
1   func PrepareExperiment(exp Experiment, hooks ...Hook) Experiment {
2       exp.Resources = CloneResources(&exp)
3       for _, route := range exp.ProductionRoutings {
4           for opName, opDesc := range route.Operations {
5               for resName, conf := range opDesc.BindTo {
6                   res, ok := exp.Resources[resName]
7                   if !ok {
8                       panic("internal error")
9                   }
10                  for _, hook := range hooks {
11                      opDesc.BindTo[resName] = hook(res, conf)
12                  }
13              }
14              route.Operations[opName] = opDesc
15          }
16      }
17      return exp
18  }
```

Листинг 2 содержит реализацию функции создания шаблона продукта.

Листинг 2 — Создание шаблона продукта

```
1   func mkProductTemplate(pd ProductionRouting, productType
        ProductTypeName) productTemplate {
2       result := productTemplate{
3           events:    make([]abstractEvent, 2*len(pd.Operations)),
4           resources: make(map[ResourceName][]bindTemplate,
                len(pd.Operations)),
5       }
6       type keydefine struct {
7           opName    OperationName
8           eventType eventType
9       }
10      dict := make(map[keydefine]index, 2*len(pd.Operations))
11      if len(pd.Operations) == 0 {
12          panic("No Operations")
13      }
14      var count int
```

```
15          keys := make([]OperationName, 0, len(pd.Operations))
16          for key := range pd.Operations {
17              keys = append(keys, key)
18          }
19          sort.Sort(byOperationNames(keys))
20          for _, opName := range keys {
21              dur := pd.Operations[opName].Duration()
22              if *dur <= 0 || dur == nil {
23                  panic("Duration <= 0!")
24              }
25              e1Ix, e2Ix := index(count*2+0), index(count*2+1)
26              count++
27              abstractOpID := abstractOperationID{productType, opName}
28              result.events[e1Ix] = abstractEvent{abstractOpID, BeginEvent,
                    nil, nil, []index{e2Ix}}
29              result.events[e2Ix] = abstractEvent{abstractOpID, EndEvent,
                    &abstractEventDef{e1Ix, *dur}, []index{e1Ix}, nil}
30              dict[keydefine{opName, result.events[e1Ix].eventType}] = e1Ix
31              dict[keydefine{opName, result.events[e2Ix].eventType}] = e2Ix
32              binds, ok := pd.Operations[opName]
33              if !ok {
34                  panic("There is no such resource")
35              }
36              for resName, conf := range binds.BindTo {
37                  result.resources[resName] = append(
38                      result.resources[resName],
39                      bindTemplate{conf, []index{e1Ix, e2Ix}},
40                  )
41              }
42          }
43
44          for _, seq := range pd.Sequence {
45              aI, aOk := dict[keydefine{seq.Next, BeginEvent}]
46              bI, bOk := dict[keydefine{seq.Prev, EndEvent}]
47              if !aOk || !bOk {
48                  panic("Error with constraints")
49              }
50
51              result.events[aI].prev = append(result.events[aI].prev, bI)
52              result.events[bI].next = append(result.events[bI].next, aI)
53          }
54          return result
55      }
```

Листинг 3 содержит реализацию функции создания заказа.

Листинг 3 — Создание заказа

```
1   func generateProductionTasks(templates
        map[ProductTypeName]productTemplate, order []OrderItem)
        []productionTask {
2       var result []productionTask
3       snCounter := map[ProductTypeName]int{}
4       for _, item := range order {
5           if item.NumberOfProducts == 0 {
6               panic("There is a product, but no NumberOfProducts")
7           }
8           template, ok := templates[item.ProductType]
9           if !ok {
10              panic("product template is not defined: " +
                    item.ProductType)
11          }
12          if _, ok := snCounter[item.ProductType]; !ok {
13              snCounter[item.ProductType] = 0
14          }
15          for i := 0; i < item.NumberOfProducts; i++ {
16              result = append(
17                  result,
18                  productionTask{
19                      productTypeName: item.ProductType,
20                      productSN:
                            ProductSN(snCounter[item.ProductType]),
21                      productTemplate: &template,
22                  },
23              )
24              snCounter[item.ProductType]++
25          }
26      }
27      return result
28  }
```

ПРИЛОЖЕНИЕ В ЭТАП ПЛАНИРОВАНИЯ

Листинг 4 содержит реализацию функции создания задания на одну единицу продукции.

Листинг 4 — Создание заданий

```
1   func (plan *ProductionPlan) appendTask(task productionTask) {
2       sn := task.productSN
3       template := task.productTemplate
4       offset := len(plan.Events)
5       for _, aEvent := range template.events {
6           op := plan.appendOperation(aEvent.abstractOperationID, sn)
7           v := ConcreteEvent{op, eventType(aEvent.eventType), nil, nil,
                nil, nil}
8           if aEvent.define != nil {
9               v.define = &concreteEventDef{nil, aEvent.define.duration}
10          }
11          plan.Events = append(plan.Events, &v)
12      }
13      for i := range template.events {
14          if plan.Events[offset+i].define != nil {
15              startIx := int(template.events[i].define.start)
16              plan.Events[offset+i].define.start =
                    plan.Events[offset+startIx]
17          }
18          for _, j := range template.events[i].prev {
19              plan.Events[offset+i].prev =
                    append(plan.Events[offset+i].prev,
                    plan.Events[offset+int(j)])
20          }
21          for _, j := range template.events[i].next {
22              plan.Events[offset+i].next =
                    append(plan.Events[offset+i].next,
                    plan.Events[offset+int(j)])
23          }
24      }
25      for resName, binds := range template.resources {
26          res, ok := plan.resources[resName]
27          if !ok {
28              panic("resource is not defined: " + resName)
29          }
30          for _, bind := range binds {
31              eventsToBind := make([]*ConcreteEvent, len(bind.indexes))
32              for i, ix := range bind.indexes {
33                  event := plan.Events[offset+int(ix)]
34                  eventsToBind[i] = event
```

```
35                      if event.EventType == BeginEvent {
36                          eventsToBind[i].resources =
                                append(eventsToBind[i].resources, res)
37                      }
38                  }
39                  res.Bind(bind.conf, eventsToBind...)
40              }
41          }
42      }
```

Листинг 5 содержит реализацию функции планирования.

Листинг 5 — Планирование

```
1       func defineProductionPlan(plan *ProductionPlan, selector
            func([]*ConcreteEvent) *ConcreteEvent) *ProductionPlan {
2           var err error
3           for err == nil {
4               err = defineNextEventGroup(plan, selector)
5           }
6           return plan
7       }
8       func defineNextEventGroup(plan *ProductionPlan, selector
            func([]*ConcreteEvent) *ConcreteEvent) error {
9       events := possibleNextEvents(plan, false)
10      events = filterPossibleEventsByResources(events)
11      if len(events) == 0 {
12          return fmt.Errorf("can't find any possibleNextEvents")
13      }
14      event := selector(events)
15      // fmt.Printf("Order:%s \n", event.Operation.Name)
16      // fmt.Printf("%v - current\n", event)
17      var newValue int64
18      for _, v := range event.prev {
19          if newValue < *v.Value {
20              newValue = *v.Value
21          }
22      }
23      for _, res := range event.resources {
24          v, isAllow := res.Constrain(event)
25          if !isAllow {
26              panic("event is not allow to process!!!")
27          }
28          if newValue < v {
29              newValue = v
30          }
31      }
```

```
32      event.SetValue(newValue)
33      doneResources(event)
34
35      for _, nextEvent := range event.next {
36          if nextEvent.define != nil && nextEvent.define.start == event {
37              newValue := nextEvent.define.duration + *event.Value
38              nextEvent.SetValue(newValue)
39              doneResources(nextEvent)
40          }
41      }
42      return nil
43  }
```

Листинг 6 содержит реализацию случайного перебора во время имитационного моделирования.

Листинг 6 — Случайный перебор

```
1       func MkExperimentWithRandom(exp Experiment, conf Config)
            []*ProductionPlan {
2           var Plans []*ProductionPlan
3
4           experimentChanel := make(chan *ProductionPlan, conf.NumOfExp)
5
6           for i := 0; i < conf.NumOfExp; i++ {
7               go func() {
8               copyExp := CopyExperimentWithRandom(&exp,int64(i))
9               experimentChanel <- MkExperiment(copyExp)
10              }()
11
12          }
13          for i := 0; i < conf.NumOfExp; i++ {
14              Plans = append(Plans,<-experimentChanel)
15          }
16          log.Println("Done")
17          file, err := os.Create("test.txt")
18
19          if err != nil{
20              panic(err)
21          }
22          defer file.Close()
23
24          value, _ := highScore(Plans)
25          if conf.Score != nil {
26              return scoreSelector(value, *conf.Score, Plans, file)
27          }
28          return Plans
```

```
29          }
```

Листинг 7 — Тестирование случайным перебором в процессе планирования

```
1   func TestMkExperimentWithRandom(t *testing.T) {
2       exp := imcore.Experiment{
3           Name: "order1",
4           ProductionRoutings:
                map[imcore.ProductTypeName]imcore.ProductionRoute{
5           "LokomotivA": {
6               Operations:
                    map[imcore.OperationName]imcore.OperationDesc{
7               "op1": imcore.NewOperationDbg(9,
                    imcore.BindConfs{"mechan":
                    resource.DemandForStaff{Amount: 4}}),
8               "op2": imcore.NewOperationDbg(8,
                    imcore.BindConfs{"mechan":
                    resource.DemandForStaff{Amount: 3}}),
9               "op3": imcore.NewOperationDbg(7,
                    imcore.BindConfs{"mechan":
                    resource.DemandForStaff{Amount: 2}}),
10              "op4": imcore.NewOperationDbg(6,
                    imcore.BindConfs{"mechan":
                    resource.DemandForStaff{Amount: 1}}),
11              "op5": imcore.NewOperationDbg(5,
                    imcore.BindConfs{"mechan":
                    resource.DemandForStaff{Amount: 4}}),
12              "op6": imcore.NewOperationDbg(4,
                    imcore.BindConfs{"mechan":
                    resource.DemandForStaff{Amount: 3}}),
13              "op7": imcore.NewOperationDbg(3,
                    imcore.BindConfs{"mechan":
                    resource.DemandForStaff{Amount: 2}}),
14              "op8": imcore.NewOperationDbg(2,
                    imcore.BindConfs{"mechan":
                    resource.DemandForStaff{Amount: 1}}),
15              "op9": imcore.NewOperationDbg(1,
                    imcore.BindConfs{"mechan":
                    resource.DemandForStaff{Amount: 1}}),
16              },
17              Sequence: []imcore.PrevAndNext{{Prev: "op1", Next:
                    "op2"},
18                  {Prev: "op1", Next: "op3"},
19                  {Prev: "op1", Next: "op4"},
20                  {Prev: "op1", Next: "op5"},
21                  {Prev: "op2", Next: "op6"},
```

```
22                          {Prev: "op3", Next: "op6"},
23                          {Prev: "op3", Next: "op7"},
24                          {Prev: "op4", Next: "op7"},
25                          {Prev: "op4", Next: "op8"},
26                          {Prev: "op5", Next: "op8"},
27                          {Prev: "op6", Next: "op9"},
28                          {Prev: "op7", Next: "op9"},
29                          {Prev: "op8", Next: "op9"},
30                      },
31                  },
32              },
33              OrderItems: []imcore.OrderItem{
34                  {ProductType: "LokomotivA", NumberOfProducts: 30},
35              },
36              Resources: map[imcore.ResourceName]imcore.Resource{
37                  "mechan": &resource.Staff{Total: 4},
38                  "DEBUG":  imcore.DebugResource{},
39              },
40              EventSelector: imcore.SimpleEventSelector,
41          }
42
43          i := 1
44          p := &i
45          plans := imcore.MkExperimentWithRandom(exp,
                imcore.Config{NumOfExp: 150, Score: p})
46          fmt.Println(plans)
47      }
```