**Введение**

В современных экономических условиях предприятия заинтересованы в сокращении издержек. С целью увеличения прибыли компании доходы инвестируются в модернизацию технологий и внедрение информационных систем. Для совершенствования управления отраслями и отдельными предприятиями на основе применения математических методов, современных средств вычислительной техники и средств связи для наилучшего использования производственных фондов, увеличения выпуска продукции, снижения ее себестоимости, повышения производительности труда, рентабельности производства и роста прибылей, то есть с целью оптимизации производства, на металлургических предприятиях внедряются системы управления производственными процессами (MES), системы планирования ресурсов предприятия (ERP), управления складами (WMS).

С целью более эффективного обеспечения безопасности данных и сохранности коммерческой тайны компании под данные информационные системы разрабатывают собственные центры обработки данных, предназначенные только для внутреннего корпоративного пользования.Такие вычислительные ресурсы нуждаются в управлении. Если на уровне автоматизированных систем управления предприятием для решения данного вопроса используется такие подходы, как контейнеризация и оркестрация, т.е. автоматическое распределение, масштабирование программных компонентов и балансировка нагрузки на них, то на уровне автоматизированных систем управления техническим процессом применяются устоявшиеся технические решения, и программное обеспечение размещается на серверах без применения контейнеризации. Однако, такие ресурсы также нуждаются в управлении, и вручную такие задачи, как, например, оптимизация размещения приложений в серверной инфраструктуре, становится решать крайне трудно и даже опасно, особенно в условиях непрерывно функционирующего предприятия с большим уровнем автоматизации.

В этой связи заслуживают изучения способы оптимизации автоматического развертывания микросервисных приложений на северах ЦОД для минимизации количества задействованных машин, что и определяет актуальность данной работы.

Объектом исследования является инфраструктура развертывания программного обеспечения на серверах центра обработки данных АО «Северсталь-инфоком».

Предметом исследования являются методы оптимизации автоматического развертывания микросервисных приложений на серверах.

Цель работы: снижение количества задействованных виртуальных машин на серверах центра обработки данных путем перераспределения развернутых на них программных компонентов.

Для достижения цели выпускной квалификационной работы необходимо выполнить следующие задачи:

1. Проанализировать существующие решения по развертыванию ПО на серверных фермах. Провести патентный обзор.
2. Рассмотреть существующие алгоритмы оптимизации, разработать математическую модель системы, выбрать наиболее подходящий алгоритм оптимизации и реализовать симулятор системы.
3. Реализовать программное обеспечение оптимизации развертывания ПО, провести экспериментальные исследования работы системы.
4. Выполнить технико-экономическое обоснование проекта.

Научная новизна работы заключается в применении генетического алгоритма при решении задачи оптимального распределения программного обеспечения на серверах и разработке функций этого алгоритма:

* первоначального заполнения;
* скрещивания;
* мутации;
* функции приспособленности.

# ГЛАВА 1. ОБЗОР СОСТОЯНИЯ РАССМАТРИВАЕМОГО ВОПРОСА.

## 1.1 Описание объекта исследования, обзор существующих решений и методов оптимизации развертывания ПО на серверах.

Главной особенностью современного этапа развития техники, в частности средств производства, является широкое использование вычислительной техники для автоматизации процессов умственного и физического труда. Коренным образом изменяется характер средств производства, по существу, создается новая материально-техническая база общества. Современная автоматизация производства объединяет множество самых разных задач - от технических до управленческих. Попытка систематизировать эти задачи привела к появлению так называемой "пирамиды автоматизации" (рисунок 1). Это модель, объединяющая все сферы деятельности современного предприятия в единую информационную среду. В основе пирамиды технологические объекты – станки, конвейеры и т.п.



Рисунок 1. Пирамида комплексной автоматизации предприятия.

В структуре пирамиды компьютерной автоматизации различают 5 уровней, связанных между собой как по горизонтали, так и по вертикали информационными каналами.

* + уровень 1 представляет из себя набор датчиков, исполнительных устройств, которые предназначены для сбора первичной информации и реализации управляющих воздействий;
  + уровень 2 содержит программируемые контроллеры, осуществляющие локальное управление технологических объектом;
* уровень 3 включает системы диспетчеризации, сбора данных и оперативного управления технологическим процессом;
* уровень 4 занимает система управления производством, позволяющая управлять производственными и людскими ресурсами в ходе технологического процесса, управлять качеством продукции, следить за обслуживанием оборудования.
* уровень 5 представляет из себя системы, оснащённые компьютерным оборудованием с программным обеспечением, позволяющим иметь полную информацию о всем производстве и осуществлять планирование ресурсов.
* уровень 6 содержит системы бизнес-аналитики. На этом уровне информация собирается в виде, удобном для восприятия и обработки.

Первые три уровня образуют АСУ ТП - автоматизированную систему управления технологическим процессом, а четвертый, пятый и шестой образуют АСУП - автоматизированную систему управления предприятием.

Работа данных информационных систем невозможна без использования центров обработки данных, которые поддерживают жизненно важные процессы. Центр обработки данных — это сложный комплекс, включающий в себя вычислительные мощности, элементы ИТ — инфраструктуры, строительных и инженерных систем, основными функциями которого являются – хранение, обработка и передача информации. В ЦОД на относительно небольшой площади сосредоточены мощные вычислительные ресурсы: сервера и системы хранения данных (СХД), осуществляющие хранение и обработку информации; сетевое оборудование, отвечающее за обмен данными внутри ЦОД, а также за связь с внешними потребителями; инженерные системы, системы безопасности, системы диспетчеризации и мониторинга, обеспечивающие эффективную работу и защиту сосредоточенного в ЦОД вычислительного центра.

При увеличивающихся темпах цифровизации возникает задача оптимального использования ресурсов серверов, т.е. уменьшения количества задействованных виртуальных машин, а, соответственно, и уменьшения потребления электроэнергии, арендной платы при учитывании ограничений, таких как необходимое количество оперативной памяти, место на жестком диске, тип процессора и других.

Рассмотрим два варианта архитектуры приложений – монолитную и микросервисную, которые активно применяются в серверном программном обеспечении. Монолитная архитектура программного обеспечения – это архитектура, в которой приложение представляется в виде одного исполняемого файла с точкой входа. Приложения, построенные с использованием такой архитектуры, отличаются такими недостатками, как сложность горизонтального масштабирования, зависимость от применения одной технологии или языка, сложность изменения кода. В свою очередь, микросервисная архитектура— вариант программного обеспечения, направленный на взаимодействие насколько это возможно небольших, слабо связанных и легко изменяемых модулей — микросервисов, лишена данных недостатков, легко масштабируется, не зависит от применения одной технологии или языка и может поддерживаться сравнительно небольшой командой разработчиков. На сегодняшний день микросервисная архитектура становится все более популярной в серверных приложениях. Современные серверные приложения могут содержать сотни микросервисов, взаимодействующих между собой с помощью сетевых протоколов.

Наряду с широким внедрением микросервисных приложений распространение получили технологии контейнеризации. Контейнеризация – это легковесная виртуализация и изоляция ресурсов на уровне операционной системы, которая позволяет запускать приложение и необходимый ему минимум системных библиотек в полностью стандартизованном контейнере, соединяющемся с хостом или чем-либо внешним по отношению к нему при помощи определенных интерфейсов. Контейнер не зависит от ресурсов или архитектуры хоста, на котором он работает, именно поэтому данная технология стала популярна.

Все компоненты, необходимые для запуска приложения, упаковываются как один образ и могут быть использованы повторно. Приложение в контейнере работает в изолированной среде и не использует память, процессор или диск операционной системы, в которой запущен контейнер. Это гарантирует изолированность процессов внутри контейнера. В результате контейнеризованное приложение может быть запущено на различных типах инфраструктуры. Благодаря такой высокой эффективности, контейнеризация обычно используется для упаковки множества отдельных микросервисов, из которых состоят современные приложения. На сегодняшний день самая распространенная система, позволяющая упаковать приложение со всем его окружением и зависимостями в контейнер – Docker.

Когда количество программных компонентов увеличивается, а процесс развертывания программного обеспечения рассматривается, как повторяющийся цикл разработки, тестирования и размещения программных продуктов на серверах, вручную решать задачу оптимального размещения программных компонентов на серверах становится не только очень сложно, но и крайне неэффективно. Появляется необходимость управлять ресурсами северов.

Отдельные контейнеры требуют координации их взаимодействия. Такая координация называется оркестрацией. Технически существует возможность обойтись без оркестрации, т.е. создать контейнер, в котором будут запущены все необходимые процессы. Однако, такой подход лишен гибкости, масштабируемости, а также возникают вопросы безопасности, поскольку запущенные в одном контейнере процессы не будут изолированы и смогут влиять друг на друга. Оркестрация позволяет создавать информационные системы из множества контейнеров, каждый из которых отвечает только за одну определенную задачу, а общение осуществляется через сетевые порты и общие каталоги. При необходимости каждый такой контейнер можно заменить другим, что позволяет, например, быстро перейти на другую версию базы данных при необходимости. Существуют различные платформы для оркестрации контейнеров. Они позволяют реализовать удобные и эффективные средства развертывания контейнерных систем, построения единой централизованной консоли для применения политик управления. Рассмотрим наиболее известные: Docker Swarm и Kubernetes. Рассмотрим их более подробно.

Docker Swarm — это система кластеризации для Docker, которая превращает набор хостов Docker в один последовательный кластер, называемый Swarm. Каждый хост в составе такого кластера выступает в качестве либо управляющего, либо рабочего узла. Запуск Docker Swarm происходит с указанием имени образа Docker, который нужно использовать для контейнеров и желаемого количества реплик. Управляющая узел автоматически назначает задачи рабочим узлам.

Docker Swarm отвечает за балансировку нагрузки и назначение уникальных DNS-имен, чтобы приложение, развернутое в кластере, можно было использовать так же, как и, если приложение было бы развернуто на одном Docker-узле, другими словами, управляющий узел распределяет запросы между рабочими узлами в кластере. Количество контейнеров может динамически масштабироваться как в сторону увеличения, так и в сторону уменьшения, управляющий узел отвечает за добавление или удаление контейнеров на узлах. Рабочие узлы постоянно контролируются управляющим и, если на каком-либо узле происходит сбой, то новые задачи запускаются на других рабочих узлах с целью обеспечения заявленного/желаемого количества копий контейнеров.

Kubernetes — система с открытым исходным кодом для управления контейнерными кластерами. Появилась в результате наработок Google при использовании механизма для изоляции процессов в виртуальной среде. Kubernetes распределяет контейнеры по узлам кластера в зависимости от текущей нагрузки и имеющихся потребностей в работе сервисов и обеспечивает распределение сетевой нагрузки. Распределение ресурсов [3] в процессе работы Kubernetes состоит из того, что ресурсы узлов динамически распределяются между выполняемыми на них контейнерами, и не нужно заботиться о том, как распределить контейнеры в кластере. Kubernetes следит за тем, чтобы не размещать на сервере больше контейнеров, чем есть ресурсов CPU для суммы потребностей всех контейнеров. Иначе, автоматически подключаются дополнительные виртуальные машины в кластер.

Балансировка сетевой нагрузки обеспечивается следующим образом. Каждое приложение разворачивается как в своем экземпляре контейнера и ему назначается IP-адрес, контейнеры скрыты за сервисом балансировки, также имеющим сетевой адрес [2]. IP-адрес сервиса используется только как точка входа и не обслуживается каким-либо процессом, слушающим этот ip-адрес и порт. В Kubernetes реализована псевдобалансировка. То есть, если запущено три контейнера, нагрузка будет распределена по следующим правилам:

1. Выбрать первый контейнер с вероятностью 33%, иначе перейти к следующему правилу.
2. Выбрать второй контейнер с вероятностью 50%, иначе перейти к следующему правилу.
3. Выбрать третий контейнер.

Такая система приводит к тому, что каждый контейнер выбирается с вероятностью 33% (Рис. 2).

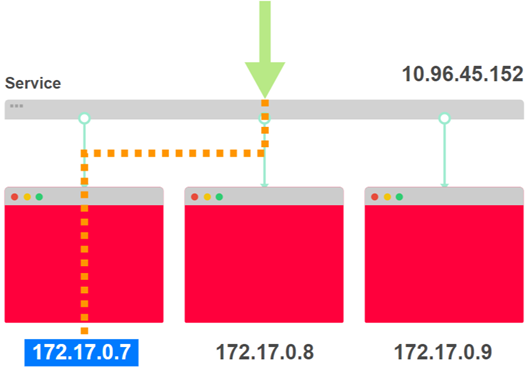


Рисунок 2. Балансировка нагрузки в Kubernetes

Использование метрик, как, например, потребность CPU в ресурсах, описанных в механизме распределения ресурсов в Kubernetes выше, может быть применено при разработке алгоритма оптимизации развертывания микросервисных приложений.

Рассмотрим способы управления динамическими рабочими нагрузками в облачных вычислениях. Облачные вычисления — модель обеспечения сетевого доступа по требованию к определенному разделяемому фонду вычислительных ресурсов, например, серверам, устройствам хранения данных, приложениям и сервиса, которые могут быть оперативно предоставлены и освобождены с минимальными эксплуатационными затратами или обращениями к провайдеру этих услуг.

Потребители облачных вычислений могут значительно уменьшить расходы на инфраструктуру информационных технологий (в краткосрочном и среднесрочном планах) и гибко реагировать на изменения вычислительных потребностей, используя свойства вычислительной эластичности облачных услуг. Одной из основных возможностей для динамического перераспределения ресурсов является живая миграция виртуальных машин [4]. Она позволяет облачным провайдерам перемещать виртуальные машины с перегруженных хостов, поддерживая их производительность при заданном SLA (соглашение об уровне услуг: средняя доступность, выраженная как среднее число сбоев на период предоставления сервиса, минимальная доступность для каждого пользователя, среднее время отклика сервиса и т.д.) и динамически консолидировать виртуальные машины на наименьшем числе хостов, чтобы экономить электроэнергию при низкой загрузке. Используя «живую» миграцию и применяя онлайн-алгоритмы, которые позволяют принимать решения о миграции в реальном времени, можно эффективно управлять облачными ресурсами, адаптируя распределение ресурсов к нагрузкам VM, поддерживая уровни производительности VM в соответствии с SLA и снижая энергопотребление инфраструктуры. Важной проблемой в контексте живой миграции является обнаружение состояния перегрузки или недогрузки хоста. Перспективным является подход принятия решений о живой миграции на основе прогнозов использования ресурсов на несколько шагов вперед. Это не только повышает стабильность, так как миграционные действия начинаются только когда нагрузка сохраняется в течение нескольких временных интервалов, но также позволяет облачным провайдерам прогнозировать состояние перегрузки до того, как это произойдёт.  Для обнаружения перегрузки используется долгосрочное прогнозирование временных рядов [5]. В контексте этой работы это означает предсказание значений для 7 временных интервалов в будущем. Хост объявляется перегруженным, если фактическое и прогнозируемое общее использование CPU для 7 временных интервалов в будущем превышают порог перегрузки. Таким же образом определяется и недостаточная загрузка сервера. Глобальный агент принимает решения о распределении ресурсов провайдера с помощью живых миграций виртуальных машин с перегруженных или недогруженных хостов на другие узлы для снижения нарушений SLA и потребления энергии. Он получает уведомления от хост-агента, если узел будет перегружен или недогружен в будущем, и выполнит перенос VM, если оно того стоит.

Рассмотренные выше технологии применяются на уровне автоматизированных систем управления предприятием. На уровне автоматизированных систем управления технологическим процессом не используется оркестрация и облачные вычисления, что обусловлено ограниченным количеством серверов, но на данном уровне все также применяются микросервисные приложения. Соответственно, необходим инструмент для решения задачи оптимального распределения полезной нагрузки на серверах.

## 1.2 Патентный обзор

Проанализируем патенты, полученные в ходе патентного поиска.

1. Управление ресурсами сервера, анализ и предотвращение вторжения к ресурсам сервера [RU 2 316 045 C2].

Сущность системы заключается в выполнении мониторинга выбранных ресурсов на одном или более компьютерных серверов и снижении коэффициента загрузки ресурсов в случае перегрузки указанных ресурсов сервера. Система предотвращения вторжения к ресурсам сервера (программный комплекс) выполняет мониторинг выбранных ресурсов на одном или более компьютерных серверах. Программный комплекс может включать один или более программный хост-компонент и программный консольный компонент. Хост-компонент располагается на компьютере сервера и отслеживает использование ресурсов сервера. В случае, если уровень загрузки конкретного ресурса превышает текущий пороговый уровень, хост-компонент может принять корректирующие меры. Указанные меры могут включать в себя, например, снижение уровня использования ресурсов или уведомление пользователя через консольный компонент. Недостатком такого подхода к управлению ресурсами состоит в жестком ограничении доступа к серверу в случае превышения порога загрузки ресурсов сервера.

1. Программно-определяемая автоматизированная система и архитектура[RU 2 729 885 C2].

Данная система предусматривает конфигурирование сети и развертывание функций/приложений автоматизации на лету на системном уровне с помощью технологий виртуализации. В исследуемом патенте описан алгоритм распределения программных компонентов. Алгоритм представляет из себя простую условную конструкцию. Компонент выбора вычислительных узлов, может использовать одно или более правил, регулирующих требования к ресурсам данного хоста, ассоциированного с гостем, чтобы выбирать вычислительный узел для развертывания. Примеры правил, которые может применять компонент выбора вычислительных узлов:

1. Если технология виртуализации хостов представляет собой виртуальную машину, то выбор вычислительного узла с высокопроизводительным процессором (например, многоядерным Xeon-процессором).
2. Если технология виртуализации хостов представляет собой контейнер, то выбор вычислительного узла со среднепроизводительным процессором (например, многоядерным Atom-процессором).
3. Если гость имеет небольшой размер (например, менее 32 MB, между 16 MB и 64 MB), то выбор вычислительного узла без программного обеспечения.

Недостатком данного алгоритма является отсутствие отслеживания загрузки каждого вычислительного узла и наиболее полного использования ресурсов каждого вычислительного узла.

1. Способ и система интеллектуального управления распределением ресурсов в вычислительных средах [RU 2 609 076 C2].

Способ управления распределением информационных ресурсов заключается в том, что посредством компьютера формируют модель использования и перераспределения ресурсов в облачных вычислительных средах. Отличительной особенностью способа является то, что модель использования и перераспределения ресурсов в облачных вычислительных средах в вычислительном облаке формируют с использованием концепции интеллектуальных алгоритмов, последовательно выполняя совокупность операций, включающую три основных этапа: на первом этапе в ВО выделяют ресурсы запускаемому экземпляру, на втором этапе проводят прогноз динамических параметров функционирования хостов ( серверов ) ВО, на третьем этапе осуществляют динамическое перераспределение ресурсов между экземплярами ВО. При этом модель формируют для максимализации показателя ЕВО эффективности функционирования ВО в соответствии с выражением (1) и учетом ограничения (2):

где – комплексный показатель эффективности ВО, отражающий совокупную синергетическую эффективность выполнения экземпляров в рамках каждого типа ресурсов ВО;

, , , – совокупность частных показателей, равных числу экземпляров с удовлетворенной потребностью в ресурсах, соответственно центрального процессора (ЦП), оперативной памяти (ОЗУ), в дисковых (Д) ресурсах, в сетевых (С) ресурсах;

– время функционирования ВО (процессорное время);

– общий объем использующихся экземплярами ресурсов ВО;

– общий объем имеющихся ресурсов ВО.

Недостатки: используется все дисковое пространство, нет стремления к минимизации использования ресурсов серверов. Такой подход может применяться при управлении всеми ресурсами вычислительного центра, но не при хостированию отдельных микросервисных приложений. Также, распределение ресурсов выполняется только на основе сформированного вычислительного облака, что приводит к увеличению времени развертывания новых экземпляров приложений.

1. Система и способ оптимизации использования ресурсов компьютера [RU 2 475 819 C1].

Согласно изобретению, дополнен следующими операциями:

1. На первом этапе совокупности операций по формированию модели производится оценка поступления ресурсов от источников, каждый источник характеризуется индексом поступления ресурсов, который задается как отношение поступивших ресурсов к запрошенным у источника.

2. На втором этапе дополнительно учитываются необходимые ресурсы для функционирующих экземпляров приложений, а также составляется перечень ресурсов, которые необходимы для экземпляров приложений, в которых имеется потребность.

3. Четвёртым этапом, на котором запрашиваются необходимые ресурсы у источников с более высоким индексом поступления ресурсов с целью получения недостающих ресурсов и последующего динамического распределения между экземплярами приложений.

Перечисленная новая совокупность существенных признаков позволяет за счет дополнительного учета характеристик источников при запросе дополнительных ресурсов не только распределять ресурсы, имеющиеся в вычислительном облаке, но и запрашивать недостающие ресурсы для вычислительного облака и выделять их экземплярам, которым не хватило ресурсов, тем самым повысить устойчивости функционирования информационно-вычислительной системы путем запуска всех необходимых экземпляров приложений в вычислительном облаке.

Недостатки: в данном изобретении не учитывается возможность нерационального использования ресурсов, то есть возможность с помощью данного алгоритма получения схемы распределения ресурсов, далекого от оптимального.

1. Система управления и диспетчеризации контейнеров [RU 2 666 475 C1].

Система управления размещением программных контейнеров предполагает, что программные контейнеры могут быть выполнены с возможностью выполнения в качестве задач в соответствии с определением задач, а определение задач может сохраняться в форме файла определения задач. Файл определения задач может описывать один или большее количество программных контейнеров, назначенных для запуска в качестве группы. Образы программного обеспечения программных контейнеров, которые могут представлять собой полную копию конкретного состояния программного контейнера на момент создания образа программного обеспечения, выполненные с возможностью выполнения в экземплярах программных контейнеров, могут предоставляться поставщику служб вычислительных ресурсов или в его местоположениях, указанных в определении задач. Определение задач также может определять потребности в ресурсах, отношения между контейнерами, используемые сетевые порты и совместно используемые ресурсы. При получении запроса на запуск задач из определения задач планировщик может определить, в соответствии со схемой размещения, какие экземпляры программных контейнеров в кластере будут выполнять задачи. В некоторых случаях поставщик служб вычислительных ресурсов может предоставлять многопользовательский планировщик для определения того, где выполнять программные контейнеры, а в некоторых случаях поставщик служб вычислительных ресурсов может позволить клиентам обеспечивать и настраивать собственные планировщики для настройки работы планировщика. В некоторых случаях планировщик может быть выполнен с возможностью использования случайной схемы выбора для случайного (согласно некоторой схеме стохастического распределения) или циклического выбора экземпляра контейнера для размещения указанного программного контейнера, учитывая потребность в ресурсах, указанную в определении задач. Данная система использует жестко заданный алгоритм распределения ресурсов, что является ее недостатком.

## 1.3 Цель и задачи работы

Обзор существующих технических средств показал, что существует достаточно решений управления развертыванием программного обеспечения на уровне автоматизации управления производством, однако, готовых систем, позволяющих оптимально хостировать микросервисные приложения на уровне автоматизированных систем управления технологическим процессом нет, что подтверждает необходимость разработки собственного решения.

Цель работы: снижение количества задействованных виртуальных машин на серверах центра обработки данных путем перераспределения развернутых на них программных компонентов.

Задачи работы:

1. Проанализировать существующие решения по развертыванию ПО на северных фермах. Провести патентный обзор.
2. Рассмотреть существующие алгоритмы оптимизации, разработать математическую модель системы, выбрать наиболее подходящий алгоритм оптимизации и реализовать симулятор системы.
3. Реализовать программное обеспечение оптимизации развертывания ПО, провести экспериментальные исследования работы системы.
4. Выполнить технико-экономическое обоснование проекта.

## 1.4 Выводы по главе 1

В данной главе были рассмотрены факторы, обуславливающие необходимость оптимизации развертывания микросервисных приложений. Были проанализированы существующие на данный момент программные продукты, позволяющие балансировать использование ресурсов и сетевую нагрузку на сервера ЦОД, однако такие средства используются на уровне автоматизированных систем управления предприятием. В ходе анализа выявлено, что готовых систем, позволяющих оптимально хостировать микросервисные приложения нет, что подтверждает необходимость разработки собственного решения.

Список литературы

1. Что такое оркестрация контейнеров <https://www.xelent.ru/blog/chto-takoe-orkestratsiya-konteynerov/>
2. Балансировка нагрузки и масштабирование соединений в Kubernetes. <https://habr.com/ru/company/mailru/blog/493820/>
3. Управление ресурсами в Kubernetes. <https://habr.com/ru/company/flant/blog/459326/>
4. Балансировка нагрузки в облаках. https://habr.com/ru/company/cloud4y/blog/329416/
5. Дориан Минаролли, Артан Мазрекай и Бернд Фрайслебен в их исследовании «Tackling uncertainty in long-term predictions for host overload and underload detection in cloud computing» <https://link.springer.com/article/10.1186/s13677-017-0074-3>
6. Jansen, K., R, O.S.: Approximation algorithms for time constrained scheduling. In: proceeding of Workshop on Parallel Algorithms and Irregularly Structured Problems, IEEE (1995) 143–157
7. Tindell, K., Burns, A., Wellings, A.: Allocating hard real-time tasks (an np-hard problem made easy). Real-Time Systems 4 (1992)
8. Monnier, Y., Beauvis, J.P., Deplanche, J.M.: A genetic algorithm for scheduling tasks in a real-time distributed system. In: Proceeding of 24th Euromicro Conference, IEEE (1998) 708–714
9. Растригин Л. А. Случайный поиск — специфика, этапы истории и предрассудки. Вопросы кибернетики. Вып. 33 (1978), с. 3–16.
10. Holland J. H. Adaptation in natural and artificial systems. Ann Arbor: University of Michigan Press. 1975.
11. Rechenberg I. Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der Biologischen Information, Freiburg: Fromman, 1973.
12. Еремеев А.В. Разработка и анализ генетических и гибридных алгоритмов для решения задач дискретной оптимизации. Дисс. канд.физ.-мат.наук. Омск, 2000.
13. Goldberg D. E. Genetic algorithms in search, optimization, and machine learning. Reading, MA: Addison-Wesley. 1989.
14. ГЕНЕТИЧЕСКИЙ АЛГОРИТМ ДЛЯ КОНКУРЕНТНОЙ ЗАДАЧИ О P-МЕДИАНЕ. Е.В. Алексеева, А.В. Орлов. Институт математики им. С.Л. Соболева СО РАН, Новосибирский Государственный Университет, Новосибирск
15. Батищев Д.И., Неймарк Е.А., Старостин Н.В. Применение генетических алгоритмов к решению задач дискретной оптимизации. Учебно-методический материал по программе повышения квалификации «Информационные технологии и компьютерное моделирование в прикладной математике». Нижний Новгород, 2007, 85 с.
16. KirkpatrickS., Gelatt C.D., Vecchi M.P. Optimization by simulated annealing // Science. 1983. Vol. 220, P.671–680.