МИНИСТЕРСТВО ПО РАЗВИТИЮ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

НУКУССКИЙ ФИЛИАЛ ТАШКЕНТСКОГО УНИВЕРСИТЕТА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ ИМЕНИ МУХАММАДА АЛ-ХОРЕЗМИ

Кафедра «Компьютер инжиниринг»

Допуск к защите

зав. кафедрой_____

2019 г. «____» _____

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

на тему «Независимая от алгоритма платформа для решения проблем, связанных с расписанием университетов»

Выпускник: З. Канназаров

Научный руководитель: д.т.н. К.Сеитназаров

НУКУС - 2019 г.

СОДЕРЖАНИЕ

Введение

Система управления проектами Project.net содержит механизм планирования, который может автоматически планировать задачи в проекте, определяя время начала и окончания каждой задачи на основе ее работы, продолжительности, назначенных ресурсов, зависимостей от других задач и других ограничений. Чтобы понять, как работает механизм планирования, полезно начать с более широкой перспективы и взглянуть на

связанные с расписанием проблемы в целом (которые включают в себя проблемы планирования проекта)

различные виды алгоритмов, используемых для генерации решений этих проблем

Цель данной статьи - предоставить вводный обзор терминологии и подходов, используемых при применении этих алгоритмов для широкого круга классических задач, связанных с расписанием. В последующих диссертациях этой серии более подробно рассматриваются различные проблемы и алгоритмы, особенно в том, что касается механизма планирования Project.net.

Integrated Computer Solutions - ведущий разработчик алгоритмов и продуктов для решения связанных с расписанием проблем, особенно используемых в механизме планирования для Project.net.

ГЛАВА I. ВВЕДЕНИЕ В АЛГОРИТМЫ ДЛЯ РЕШЕНИЯ ЗАДАЧ, СВЯЗАННЫХ С РАСПИСАНИЕМ

1.1. Проблемы с расписанием

Автоматическое планирование, при правильном использовании, является одним из самых мощных инструментов для повышения эффективности и стоимости управления проектами, планирования или производства. Планирование может использоваться для долгосрочного планирования, включающего несколько проектов, для подробного планирования, включающего задачи и задания в повседневной работе. Приложения могут варьироваться от планирования персонала в розничном магазине до управления проектами в крупной компании. На рынке имеется большое количество программного обеспечения, которое предоставляет решения для планирования задач с различными типами и конфигурациями в зависимости от его использования.

Целью автоматического алгоритма для задачи, связанной с расписанием, является создание действительно хорошего (то есть оптимального или почти оптимального) расписания для набора видов деятельности и / или для ресурсов, которые должны быть им назначены. Это может включать

Планирование: определение последовательности действий, а также времени начала и окончания каждого из них.

Назначение: назначение ресурсов (например, людей, машин) для действий

Проблемы, связанные с расписанием, можно разделить на три категории

Связанные с расписанием задачи Задачи просто сосредоточены на назначении, где последовательность и время действий заранее определены. Многие из этих проблем известны как проблемы с планированием трудовых ресурсов и обычно возникают из-за управления назначениями персонала в таких организациях, как розничные магазины и больницы, как правило, в течение

относительно короткого периода времени планирования. Эти проблемы включают планирование выходных, расписание смен и планирование экипажа.

Задачи только для планирования просто сосредоточены на определении последовательности и времени действий (обычно называемых задачами), где назначение ресурсов действиям либо предопределено, либо игнорируется. Стандартные задачи планирования проекта, которые сосредоточены на планировании задач в рамках проекта (или связанных проектов), попадают в эту категорию, включая Основное планирование проекта и Планирование проекта с ограничением ресурсов / размера. Базовое управление портфелем также является частью этой категории.

Совместное планирование / задачи назначения, где необходимо определить последовательность и время действий, а также распределение ресурсов для них. Эти проблемы включают планирование работы, расширенное управление портфелем и планирование проекта с назначением.

Проблемы, связанные с расписанием, являются частью более широкого класса проблем, называемых задачами оптимизации, которые имеют две важные особенности:

Ограничения - формальное описание требований, которые должны быть удовлетворены кандидатом на решение проблемы - например, что конкретная задача не может быть запущена, пока не завершится какая-то другая задача.

Целевые функции - математическая характеристика качества решения. В целом, целевая функция стремится минимизировать (или максимизировать) некоторую ценность, расчеты которой основаны на деталях возможного решения - например, минимизировать время, необходимое для выполнения всех действий; это обычное ограничение в задачах планирования, и более формально известно как минимизация рабочего времени.

Проблемы, связанные с расписанием, могут рассматриваться как проблемы в автономном режиме или в сети.

В автономных задачах - информация обо всех действиях, ресурсах, ограничениях и целевых функциях известна заранее, и цель состоит в том, чтобы найти единственное «хорошее» решение проблемы

В онлайн-проблемах - происходят постоянные изменения в действиях (например, задержки, отмены) и ресурсах (например, болезнь, поломка), а иногда даже в ограничениях и целевых функциях, так что появляются новые решения (для тех действий, которые еще не завершены).) должен постоянно пересчитываться.

1.2. Распределение ресурсов

В простейших задачах, связанных с расписанием, любой ресурс может быть назначен любому действию, но каждому действию назначается только один ресурс за раз, и каждый ресурс назначается только одному действию за раз, непрерывно с момента действия действия. начинается в то время, когда это заканчивается. Однако на практике мы находим более сложные формулировки проблемы, которые необходимо решить:

Ограниченные назначения: ресурс может быть назначен деятельности только для части (или для нескольких частей) продолжительности действия

Назначение нескольких ресурсов. Для действия может быть назначено несколько ресурсов.

Частичные назначения: когда ресурс назначен действию, он может потратить только часть своего доступного времени, работая над ним

Несколько назначений: ресурс может быть назначен нескольким действиям одновременно (что требует поддержки частичных назначений)

Резервирование: ресурсы могут быть зарезервированы (полностью или частично) для конкретного проекта (или группы мероприятий), что ограничивает то, как ресурс может быть назначен и / или когда могут быть запланированы действия, предварительно назначенные этому ресурсу.

Право на участие: действие может разрешать назначать ему только определенные ресурсы или типы ресурсов.

Вытеснение / разделение: ресурс, назначенный одному действию, может быть прерван и назначен другому (например, более высокоприоритетному) действию, но затем может вернуться к продолжению работы над начальным действием.

В онлайн-задачах (особенно в планировании заданий) это называется вытеснением, поскольку идея заключается в том, что внезапно необходимо запланировать более приоритетное действие, требующее повторного назначения ему ресурса. В автономных задачах (особенно в планировании проекта) это называется разделением действий или задач.

Невозобновляемые ресурсы. Ресурсы невозобновляемых ресурсов (например, деньги, химикаты и т. Д.) Могут быть израсходованы, в отличие от людей, которых можно назначать, пока они доступны. Поэтому необходимо найти графики, которые не требуют больше ресурсов, чем доступно. Более сложные формулировки включают частично возобновляемые ресурсы и ограничения.

1.3. Оптимальные и эвристические алгоритмы

Существуют алгоритмы, которые могут найти оптимальное решение (такое, которое дает минимальное / максимальное значение целевой функции) за «разумное» время (относительно размера задачи) для самых простых задач, связанных с расписанием, - в частности, для простых версий планирования рабочей силы и основного планирования проекта.

Тем не менее, для почти всех проблем, связанных с расписанием, которые мы упоминали, оптимальные алгоритмы просто занимают слишком много времени для всех, кроме самых маленьких задач. Таким образом, для большинства из этих проблем нам нужно использовать эвристические

алгоритмы, которые могут производить решения, близкие к оптимальным, то есть достаточно близкие к оптимальному, за разумное время.

Существует ряд общих эвристических подходов к алгоритмам, используемым для решения проблем, связанных с расписанием, которые можно разделить на

Алгоритмы построения (включая алгоритмы жадности), которые начинаются с пустого или неполного решения (например, когда задачи не запланированы и / или не назначены ресурсы) и постепенно делают его более полным (например, путем планирования одной дополнительной задачи и / или назначения одной дополнительный ресурс за раз)

Алгоритмы поиска, которые начинаются с одного или нескольких законченных решений-кандидатов и постепенно объединяют и / или модифицируют их с целью создания более совершенных комплексных решений. К ним относятся подходы (также называемые метаэвристикой), такие как восхождение на холм, поиск по Табу, имитация отжига, оптимизация роя частиц, алгоритмы искусственного пчелиного семейства, генетические алгоритмы и дифференциальная эволюция.

Гибридные алгоритмы, которые используют комбинацию подходов построения и поиска

ГЛАВА II. АЛГОРИТМЫ ДЛЯ ПРОБЛЕМ ПЛАНИРОВАНИЯ КУРСА УНИВЕРСИТЕТА

2.1. Структура планировка курса университета

Эта диссертация посвящена проблеме планирования курсов, когда у нас есть набор курсов, лекторов и аудиторий. Курсы назначаются и планируются таким образом, чтобы общее предпочтение было максимальным. Разработана математическая модель задачи в виде линейной целочисленной программы. Задача небольшого размера может быть оптимально решена с помощью коммерческого программного обеспечения. Затем мы разработали три различных метаэвристики, основанные на алгоритмах искусственного иммунного, генетического и имитационного отжига. Эти три метода решения оснащены новыми процедурами, такими как операторы перемещения и пересечения. Параметры предлагаемой метаэвристики сначала настраиваются, а затем оцениваются с помощью оптимальных решений, найденных моделью. Кроме того, они оцениваются путем сравнения их показателей. Эксперименты показывают, что алгоритм искусственного иммунитета работает лучше, чем другие алгоритмы.

Радсебави проблем планирования предваряющих постов низшей коллегии, предаварий и учителей. Коллегии с доделюю и планирую тако да с максимально задоволье преференции. Развивающие математические модели проблематики у линеарного програма cijelih Brojeva. Манджи суть проблемы оптимальны для сравнения с софтвера. Затим развивать три вида метаэвристики на тему умных имуних, генетических и алгоритмических методов. Те три способа определения опережения в новом суждении о том, что операторы критания и крижанья. Параметр предлагает метаэвристику, основанную на условных обозначениях, оптимальной и оптимальной модели модели. Надежда се проредженю успешбом ніхихих перформансси. Экспертизы показывают, что я иммуни алгоритам успеваемости друг друга.

Планирование - это проблема выполнения набора заданных наборов ограниченных ресурсов. Среди важных задач календарного планирования проблема университетского планирования (США) привлекает большое внимание как искусственного интеллекта, так и исследования операций [2]. В классических задачах США набор событий (таких как курсы и экзамены) присваивается ряду временных интервалов в классе, чтобы удовлетворять набору ограничений [1].

Задачами США могут быть либо экзамен, либо планирование курса. В расписании экзаменов одной из целей является распределение нескольких экзаменов как можно более равномерно, в то время как при планировании курса студенты требуют как можно более компактного графика [3]. В этой статье рассматривается класс задач планирования университета, известный как проблема планирования университетского курса (UCS). В каждом семестре академическая служба всегда планирует сложную работу по планированию курсов.

Задача UCS является NP-сложной [4] и включает в себя два решения: назначение инструктора и планирование класса. В назначении инструктора мы определяем, какой преподаватель будет представлять какие курсы. В расписании занятий мы указываем, в каком классе будет представлен каждый курс. Решения должны быть приняты таким образом, чтобы выполнить ряд ограничений. Он учитывает профессиональную квалификацию инструктора, предпочтения относительно курсов и дней, разумное распределение сверхурочной работы среди инструкторов и наличие оборудования и средств. Кроме того, это позволяет избежать любого конфликта в расписаниях инструкторов и учебных аудиториях.

Ограничения задачи UCS можно разделить на две группы; твердый и мягкий [5]. Пока все жесткие ограничения выполнены и выполнены, график выполним. Типичным примером жестких ограничений является то, что ни один преподаватель не может преподавать не более одного курса за раз. С другой

стороны, мягкие ограничения - это те требования, которые хотя и необходимы, но должны быть выполнены в максимально возможной степени. То есть они могут быть нарушены при необходимости. В результате мягкие ограничения могут упоминаться как предпочтения и использоваться для оценки качества решения. Например, типичным мягким ограничением является распределение классов как можно более равномерно. Основная цель - найти выполнимый график, удовлетворяющий всем жестким ограничениям и максимизирующий удовлетворение как преподавателей, так и классов на основе их предпочтений. Это равносильно минимизации нарушения мягкого ограничения.

Проблемы UCS обычно отличаются от одного университета к другому [2, 6]. Весьма вероятно, что каждый университет имеет свой уникальный набор требований для эффективного использования своих ресурсов, выполнения требований своего бизнеса, обеспечения высокого уровня удовлетворенности своих студентов и так далее. Следовательно, для удовлетворения всех этих уникальных требований должна быть разработана система планирования курсов.

Различные аспекты проблемы впервые описаны. Математическая постановка задачи затем представляется в виде целочисленной линейной программы. Используя указанное программное обеспечение математического программирования, модель решена. В последнее время интерес к метаэвристике для решения задач UCS растет. Такие алгоритмы включают поиск по табу Лю и Хао [7], имитацию отжига по Жангету и др. [8] и генетический алгоритм Ван [9]. Мы предлагаем три метаэвристики, основанные на алгоритмах искусственного иммунного, генетического и имитированного отжига, чтобы решить эту проблему в больших случаях. Эти алгоритмы используют новые процедуры, такие как операторы перемещения и пересечения.

Остальная часть этой статьи организована следующим образом. В разделе 2 проблема математически сформулирована. В разделе 3 разрабатываются

алгоритмы решения. В разделе 4 проводятся эксперименты. Наконец, в разделе 5 документ заканчивается.

л - количество инструкторов

в - количество курсов

r - количество классных комнат

d - количество дней

t - индекс рабочих дней {1, 2,…, d}

i - индекс для инструкторов, где {1, 2,…, l}

j - Индекс для курсов, где {1, 2,…, c}

k - Индекс для классных комнат {1, 2,…, e}

h - индекс за период времени {1, 2, 3}

1, Jiu- Полезность инструктора я для преподавания курса J

2, тиу- Полезность инструктора я для преподавания курса т

3, тью- Полезность инструктора J для преподавания курса Т

gi, t - 1, если инструктор меня может пригласить в день t, и 0 в противном случае

si, j - 1, если инструктор, я могу преподавать курс j, и 0 в противном случае

vj, k - 1, если курс j может быть представлен в классе k, и 0 в противном случае

Decision variables:

$X_{i,j,t,j,h}$ - 1 if instructor i teaches course j on day t in classroom l in time period h, and 0 otherwise

$Y_{i,t}$ - Binary variable taking value 1 if instructor i is invited on day t, and 0 otherwise.

The model is as follows:

$$\text{Maximize } Z = \sum_{i=1}^{l}\sum_{j=1}^{c}\sum_{t=1}^{d}\sum_{k=1}^{e}\sum_{h=1}^{3} X_{i,j,t,k,h} \cdot u_{i,j}^{1} + \\ + \sum_{i=1}^{l}\sum_{t=1}^{d} Y_{i,t} \cdot u_{i,t}^{2} + \sum_{i=1}^{l}\sum_{j=1}^{c}\sum_{t=1}^{d}\sum_{k=1}^{e}\sum_{h=1}^{3} X_{i,j,t,k,h} \cdot u_{j,t}^{3} \quad (1)$$

Subject to:

$$\sum_{i=1}^{l}\sum_{t=1}^{d}\sum_{k=1}^{e}\sum_{h=1}^{3} X_{i,j,t,k,h} = 1 \qquad \forall_{j} \qquad (2)$$

$$\sum_{j=1}^{c}\sum_{k=1}^{e} X_{i,j,t,k,h} \le Y_{i,t} \qquad \forall_{i,t,h} \qquad (3)$$

$$Y_{i,t} \le g_{i,t} \qquad \forall_{i,t} \qquad (4)$$

$$\sum_{i=1}^{l}\sum_{j=1}^{c} X_{i,j,t,k,h} \le 1 \qquad \forall_{t,k,h} \qquad (5)$$

$$\sum_{t=1}^{d}\sum_{k=1}^{e}\sum_{h=1}^{3} X_{i,j,t,k,h} \le s_{i,j} \qquad \forall_{i,j} \qquad (6)$$

$$\sum_{i=1}^{l}\sum_{t=1}^{d}\sum_{h=1}^{3} X_{i,j,t,k,h} \le v_{j,k} \qquad \forall_{j,k} \qquad (7)$$

$$Y_{i,t} \in \{0,1\} \qquad \forall_{i,t} \qquad (8)$$

$$Z_{i,j,t,k,h} \in \{0,1\} \qquad \forall_{i,j,t,k,h} \qquad (9)$$

Eq. (1) рассчитывает общую полезность. Набор ограничений (2) гарантирует, что каждый курс преподается. Набор ограничений (3) указывает дни, когда инструктор должен преподавать курсы. Более того, это гарантирует, что каждый преподаватель в каждом временном интервале будет представлять не

более одного курса. Набор ограничений (4) обеспечивает приглашение инструкторов в те дни, которые они предпочитают. Набор ограничений (5) предотвращает перекрестное назначение, то есть в каждом классе одновременно может быть представлено не более одного курса. Набор ограничений (6) должен гарантировать, что каждый инструктор назначен на курсы экспертизы. Набор ограничений (7) указывает, что каждый курс назначается подходящим классам. Наборы ограничений (8) и (9) определяют решающие двоичные переменные.

2.2. Алгоритмы решения

Поскольку проблема NP-трудна, наиболее эффективными алгоритмами для ее решения являются метаэвристика. Примеры этих алгоритмов включают эвристику раскраски графов [2], поиск Tabu [7], имитационный отжиг [8], эволюционные алгоритмы [10], рассуждения на основе случая [11], двухэтапные эвристические алгоритмы [12], поиск табу [13], муравьиная колония [14] и так далее. Заинтересованные читатели могут обратиться к [6] за всесторонним обзором автоматизированных подходов к учету времени в университетах, представленным в последние годы. В этой статье предлагаются три различных метода метаэвристики: искусственный иммунный, генетический и имитационный отжиг. Сначала мы объясним схему кодирования, используемую в алгоритмах, а затем опишем алгоритмы.

2.3. Схема кодирования

Первым шагом в разработке алгоритма является разработка схемы кодирования для представления решений проблемы. Мы представляем пространство решений двумя двоичными матрицами и правилом диспетчеризации. Первая двоичная матрица показывает назначение курса лектора; то есть, какой курс преподает какой лектор. В этой матрице строки и столбцы представляют курсы и лекторов, соответственно, где «1» означает назначение, в то время как «0» означает отсутствие назначения, а «-» означает, что соответствующий лектор не может преподавать курс.

Вторая матрица представляет приглашение дня лектора; то есть каждый курс-лектор представлен в какой день. В этой матрице строка и столбцы представляют дни и лекторов, соответственно, где «1» означает приглашение, «0» означает отсутствие приглашения, а «-» означает, что лектор не может быть приглашен в этот день. Обратите внимание, что лектор на каждый день может иметь максимум 3 курса. Поэтому количество дней, которые приглашает лектор, зависит от количества назначенных курсов.

Применяемое здесь правило диспетчеризации заключается в назначении курсов для временных интервалов в классе. Раз два решения, конечно,

задание лектора и день лектора указаны, решение о временном интервале в классе остается; то есть, какой курс представлен в каком классе и какие временные интервалы касаются жестких ограничений проблемы. Для этого мы предлагаем следующее правило. Каждый курс назначается первому доступному классу, который подходит для курса, когда преподаватель также доступен. Если лектора приглашают более чем на один день, каждый курс представляется в день с наивысшим предпочтением и доступными классными комнатами. Обратите внимание, что это представление является полным и косвенным. Это является косвенным, поскольку нам нужно декодировать решение, чтобы вычислить целевые функции, и оно завершено, поскольку все возможные решения для проблемы могут быть представлены.

Схема кодирования описывается посредством применения к иллюстративному примеру. Рассмотрим проблему с восемью курсами, четырьмя преподавателями, двумя рабочими днями и двумя классными комнатами с четырьмя временными интервалами на каждый день. Кодированное решение представлено на рис. 1. На этом рисунке часть A показывает первую матрицу, а часть b - вторую матрицу. В этом решении курсы 1 и 6 назначаются лектору 1, курсы 3 и 7 - лектору 2, а курсы 2, 4, 5 и 8 - лектору 4. Курс лектора 3 не

назначается. Лекторы 1 и 4 приглашаются на первый день и лекторы 2 и 4 на второй день. Поэтому Лектор 4 приглашается в оба дня.

Course \ *Lecturer*	1	2	3	4
1	1	-	-	0
2	-	0	-	1
3	-	1	0	-
4	-	-	-	1
5	0	-	0	1
6	1	-	0	-
7	0	1	-	-
8	-	-	0	1

Day \ *Lecturer*	1	2	3	4
1	1	0	-	1
2	-	1	0	1

b) Instructor-day assignment

Figure 1 An example of encoded solution.

б) день инструктора

Рисунок 1 Пример закодированного решения.

2.4. Генетический алгоритм

Генетический алгоритм (GA) предназначен для решения некоторых проблем промышленности, которые было трудно решить с помощью

традиционных методов. Сегодня GA - это известные популяционные эволюционные алгоритмы, решающие как дискретные, так и непрерывные задачи оптимизации. Идея, лежащая в основе GA, исходит из концепции «выживания наиболее приспособленных» Дарвина, означающей, что хорошие родители дают лучших потомков.

2.5. Общая структура

GA ищет пространство решений с популяцией хромосом, каждая из которых представляет закодированное решение. Значение пригодности присваивается каждой хромосоме в соответствии с ее характеристиками. Чем лучше хромосома, тем выше становится это значение. Население эволюционирует с помощью набора операторов, пока какой-либо критерий остановки не будет посетил. Типичная итерация генерации GA происходит следующим образом. Лучшие хромосомы современной популяции непосредственно копируются в следующее поколение (репродукция). Механизм отбора выбирает хромосомы текущей популяции, чтобы дать более высокий шанс хромосомам с более высоким значением пригодности. Отобранные хромосомы пересекаются, чтобы произвести новое потомство. После процесса скрещивания каждое потомство может мутировать другим механизмом, называемым мутацией. После этого новая популяция оценивается снова и весь процесс повторяется. Контур предлагаемого ГА приведен на рис. 2.

The procedure: the proposed GA
Initialization mechanism
While *the stopping criterion is not met* **do**
    Selection mechanism
    Crossover mechanism
    Mutation mechanism
End while

Рисунок 2 Схема предлагаемого ГА

2.5. Механизмы инициализации и выбора

GA начинается с ряда хромосом, каждая из которых представляет возможное решение. Количество хромосом - это размер популяции, указанный поп. Исходные хромосомы случайным образом генерируются из возможных решений.

После инициализации алгоритмов оценивают каждую хромосому и определяют ее пригодность (то есть целевую функцию). Вероятность выбора хромосомы k для механизма кроссовера заключается в следующем.

$$p_k = \frac{fit(k)}{\sum_{h=1}^{pop} fit(h)}, \qquad (10)$$

где fit (k) - приспособленность хромосомы k.

Механизмы кроссовера и мутации

Новые решения создаются путем скрещивания двух других родителей с помощью оператора, называемого кроссовером. Обратите внимание, что операторы кроссовера должны избегать создания недопустимых решений. Цель состоит в том, чтобы произвести лучшие потомства. Чтобы найти решение для лучших областей, мы определяем новое решение, которое наследуется от обоих родителей. Фактически, мы объединяем двух родителей, чтобы сформировать новое решение. Это делается через оператора с помощью следующих шагов.

Генерируется случайное число, равномерно распределенное между 0 и 1. Если это число меньше 0,6, два столбца этого инструктора от родителя 1 (то есть один из назначения курса инструктора и один в назначении дня инструктора) копируются в новое решение. Другие столбцы нового решения заполняются из родительского 2. Обратите внимание, что новое решение, вероятно, нуждается в

модификации, чтобы быть возможным решением. При назначении курса для инструктора, если курс назначается двум инструкторам, один из инструкторов выбирается случайным образом, а другой вычеркивается. Более того, если курс не назначен ни одному инструктору, он случайным образом назначается лектору.

Что касается этого нового назначения для инструктора, то его назначение на день инструктора обновляется.

После кроссовера каждое решение изменяется оператором мутации. Основная цель применения мутации состоит в том, чтобы избежать сближения с локальным оптимумом и диверсифицировать население. Мы используем следующий оператор мутации. Один преподаватель выбирается случайным образом, и затем день приглашения меняется с дня с наибольшей загрузкой курса на день с наименьшей нагрузкой курса. Второй локальный поиск применяется ко всем инструкторам в произвольном порядке без повторений. Следовательно, это приводит к появлению mm новых решений, созданных путем перепланировки каждого инструктора. Лучшее решение среди этих новых решений выбрано.

Имитация отжига

Имитация отжига (SA) - это метаэвристика, основанная на локальном поиске, моделирующая процесс отжига [15]. SA включает в себя механизм, называемый критерием принятия, который позволяет ему выйти из локального оптимума. Критерий принятия определяет, следует ли принять или отклонить новое сгенерированное решение. В этом механизме могут быть приняты даже худшие решения.

Имитация отжига обрабатывает исходное решение и делает серию ходов, пока не будет достигнут критерий остановки. Идея состоит в том, чтобы сгенерировать новое решение s оператором из текущего решения x. Другое случайное правило используется для принятия или отклонения этого нового решения. Правило принятия контролируется параметром t, называемым температурой. Разброс между объективными значениями двух возможных

решений рассчитывается Δ = fit (s) - fit (x). Если Δ ≤ 0, решение s принимается. В противном случае решение *ss* принимается с вероятностью, равной exp (Δ / ti). При каждой температуре ti алгоритм работает с фиксированным числом ходов. Пока SA продолжается, температура постепенно снижается. Используется график экспоненциального охлаждения, ti = α · ti-1 (где α ∈ (0, 1) - скорость снижения температуры). Начальная температура установлена равной 50 и α = 0,97. На рис. 3 показан общий план предлагаемой СА.

The procedure: the proposed SA
Initialization mechanism
While *the stopping criterion is not met* **do**
Move mechanism
Acceptance mechanism
Cooling schedule mechanism
End while

Figure 3 The outline of the proposed SA

Рисунок 3 Схема предлагаемой SA

Оператор перемещения

В этой статье мы генерируем новое решение из текущего решения по следующей процедуре. Мы выбираем один курс случайным образом и назначаем его другому квалифицированному инструктору. Обратите внимание, что переназначение курса влияет на двух преподавателей, один с сокращением курса, а другой с добавлением курса. Следовательно, количество дней, которое приглашают эти два инструктора, может потребоваться обновить. Учитывая это новое задание, приглашения инструкторов на день инструктора обновляются.

Искусственный иммунный алгоритм

Алгоритм искусственного иммунитета (AIA) - это метаэвристический метод, основанный на популяции, который означает, что он проводит параллельный поиск по нескольким закодированным решениям. AIA известен как эффективный алгоритм [16, 17]. Основная идея исходит из моделирования

физиологической иммунной системы природных живых организмов. Иммунная система защищает организм от инородных вирусов (антигенов). Это делается с помощью набора антител. В случае обнаружения антигена антитела, идентифицирующие этот антиген, размножаются путем клонирования. То есть всякий раз, когда антиген проникает в организм, они сначала распознают, что эти антитела более подходят для борьбы с вторжением антигена. Затем система производит больше вариаций этих антител в следующем поколении.

Качество каждого антитела показано значением сродства. Антитело с большей аффинностью может лучше бороться с антигенами. Антиген относится к рассматриваемой проблеме. Каждое антитело является возможным решением, а сродство является значением целевой функции, полученным антителом.

Общая структура

Процедура AIA может быть описана следующим образом. AIA ищет проблемное пространство с популяцией антител. Значение сродства присваивается каждому антителу в соответствии с его эффективностью. Чем более желательным является антитело, тем выше становится это значение. Используя иммунных операторов, таких как отбор клонирования и созревание аффинности, из текущей популяции генерируется новая популяция. Отбор клонирования относится к пролиферации антител, которые лучше устраняют антигены.

Механизм созревания аффинности состоит из гипермутации и редактирования рецептора. Гипермутация соответствует созданию новых решений, похожих на их создателей, но не совсем одинаковых. Низшие антитела должны быть подвергнуты гипермутации с более высокой скоростью, тогда как более высокие антитела подвергаются более низкой скорости. Редактирование рецептора - это механизм определения скорости гипермутации. AIA действует следующим образом. Определяемая пользователем аффинная функция рассчитывает количество клонов, пролиферированных от каждого антитела. Все

пролиферированные клоны собираются в мутирующий пул. Клоны мутирующего пула подвергаются гипермутации для генерации новых антител. Затем оценивается новая популяция и весь процесс повторяется. Исходные хромосомы случайным образом генерируются из возможных решений.

Процедура выбора клонирования

Аффинность измеряет качество каждого антитела для устранения антигенов. Чем выше значение сродства, тем лучше антитело в борьбе с антигенами. Мы можем предположить нашу проблему оптимизации как антиген и закодированное решение как антитело. Лучшими антителами являются те, которые достигают лучшей цели (то есть они могут лучше бороться с проблемой оптимизации). Мы устанавливаем сродство каждого антитела, равное значению целевой функции.

Вероятность клонирования для каждого антитела в мутантный пул является пропорцией его значения аффинности. Таким образом, антитела с более высокой аффинностью более вероятны иметь больше клонов. Мутирующий пул имеет фиксированный размер клонов. Одно из них выполнено лучшим антителом из текущей популяции. Для выполнения остального мы используем механизм отбора. Шанс антитела рассчитывается по формуле (10).

Процедура созревания сродства

Основной функцией процедуры созревания аффинности является случайное изменение всех клонов, существующих в пуле. Каждый клон подвергается различной скорости изменения в соответствии со значением сродства. Низшие клоны подвергаются высокой степени изменения, в то время как лучшие клоны претерпевают небольшие изменения. Эти изменения вносятся оператором, называемым гипермутацией. В качестве гипермутации низкой скорости мы используем следующий оператор.

Один курс выбирается случайным образом и перераспределяется на другого квалифицированного лектора, который имеет наибольшее предпочтение.

Обратите внимание, что переназначение курса затрагивает двух лекторов, одного с сокращением курса, а другого с добавлением курса. Таким образом, количество дней, которое приглашает каждый из этих двух лекторов, может измениться. Что касается этого нового задания, обновляются приглашения на день лектора для обоих лекторов. Локальный поиск повторяется для всех курсов наугад без повторения. Таким образом, n новых решений генерируется путем переназначения каждого курса. Лучшее решение среди этих новых решений выбрано.

В качестве высокоскоростной гипермутации мы используем оператор мутации, определенный в GA. Должен быть определен критерий для определения условия, при котором используется один из операторов. Мы используем гипермутацию низкой скорости, если мы имеем

$$\frac{affinity(x) - affinity(best\ antibody)}{affinity(best\ antibody)} < 0.1$$

В противном случае применяется высокоскоростная гипермутация.

В отличие от предыдущей работы [16], в которой дети принимаются только в том случае, если они имеют меньший рабочий период, чем их создатели, мы используем простой критерий приемлемости, подобный отжигу. Помимо принятия лучшего потомства, худшее потомство может быть принято с вероятностью

$$\exp\left(\frac{affinity(x) - affinity(creator)}{20}\right) < 0.1$$

При этом мы позволим алгоритму легко избежать застревания в локальных минимумах. Нам необходимо заявить, что мы применяем элитную стратегию, то есть лучший клон напрямую копируется в следующее поколение.

В этом разделе мы оцениваем производительность модели и предлагаемых алгоритмов. Для этого генерируются два набора экспериментальных примеров. Первый набор состоит из экземпляров небольшого размера и используется для оценки способности модели решать проблему, а также общей производительности алгоритмов. Мы рассмотрим следующие небольшие размеры для (m, n):

(20, 5), (20, 7), (40, 10), (40, 15), (60, 10), (60, 15), (80, 15), (80, 20), (100 , 20) и (100, 25)

Второй набор включает в себя экземпляры большого размера, с помощью которых сравнивается производительность предложенных алгоритмов. Рассмотрим следующие 6 комбинаций для (m, n):

(100, 20), (100, 30), (200, 30), (200, 50), (300, 50) и (300, 70)

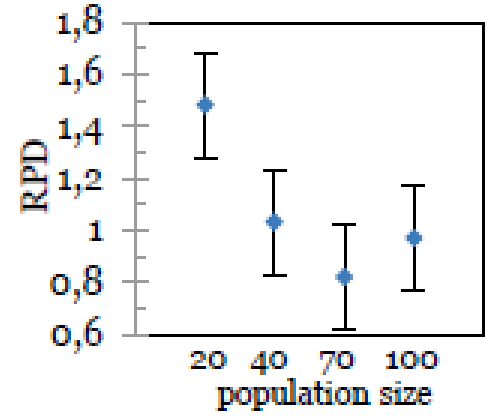
Предполагается также, что существует 0,2 млн. Классных комнат, в неделю по 5 рабочих дней и 3 временных интервала на каждый день. Мы генерируем 10 экземпляров для каждого размера задачи путем случайного генерирования других параметров задачи. Необходимо соблюдать осторожность при создании данных. Например, чтобы преподавать каждый курс, должен быть хотя бы один преподаватель.

Модель и алгоритмы реализованы в CPLEX и C ++ соответственно. Они работают на ПК с 2,0 ГГц процессором Intel Core 2 Duo и 2 ГБ оперативной памяти. Модель допускается максимум 600 секунд вычислительного времени. Критерий остановки для алгоритмов устанавливается на ограничение времени ЦП, равное нм секундам. Этот критерий остановки дает больше времени по мере увеличения количества курсов или лекторов. Относительное процентное

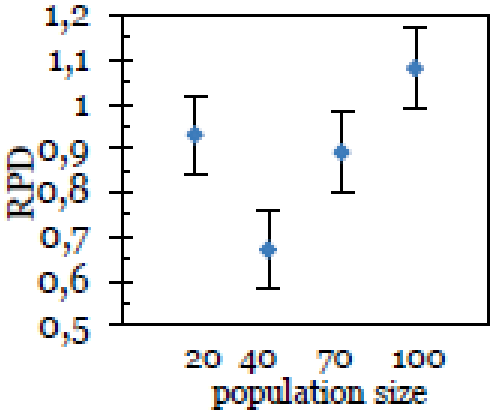
отклонение (RPD) используется в качестве показателя эффективности [18]. RPD рассчитывается следующим образом.

$$RPD = \frac{ALG - Max}{Max} \times 100$$

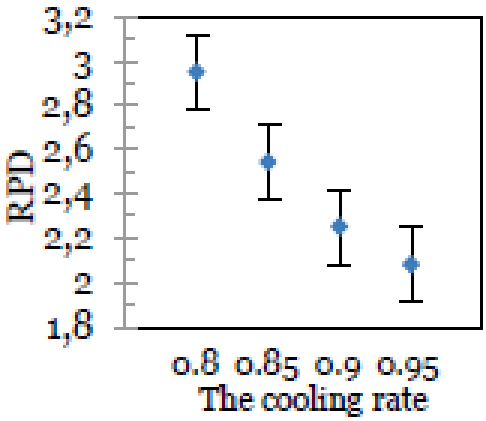
где $AAAAAA$ и MMM - решение алгоритма и верхняя граница данного экземпляра, которая равна наилучшей цели, найденной для данного экземпляра



a) GA



b) AIA



c) SA

Рисунок 4 Средние интервалы RPD и LSD для протестированных алгоритмов

Установка параметров

Параметр GA и AIA - это численность населения, а SA - скорость охлаждения. Рассмотренные размеры популяции составляют {20, 40, 70, 100}. Рассматриваемые уровни скорости охлаждения: {0,95, 0,90, 0,85, 0,8}. 20 разных экземпляров генерируются. Затем мы решаем их по полученным алгоритмам. Рис. 4 показывает результаты. Как можно видеть, для GA лучший размер популяции составляет 70, тогда как для AIA это значение равно 40. Лучшая скорость охлаждения также составляет 0,95.

2.7. Эксперимент на небольших размерах

В этом подразделе мы решаем мелкие экземпляры с помощью модели и алгоритмов. Tab. 1 представляет результаты. Модель способна решать случаи до m = 60 и n = 15 менее чем за 60 секунд. Экземпляры с m = 80 требуют менее 500 секунд. Модель не может решить случаи с m = 100 менее чем за 600 секунд.

Tab. 2 показывает среднее значение RPD, полученное с помощью проверенных алгоритмов для каждого размера группы. Как видно, AIA превосходит другие алгоритмы со средним RPD 0,87%. GA получил второе место с 1,47%. Наихудший алгоритм - SA со средним RPD 1,91%. Анализируя производительность по сравнению с различными размерами проблем, предлагаемая AIA хорошо работает во всех размерах.

Table 1 The model's results on small-sized instances

Instance			Optimality gap	Computational time (sec)
m	*n*	*e*		
20	5	4	0	0,6
20	7	4	0	1,4
40	10	5	0	42
40	15	5	0	5
60	10	6	0	92
60	15	6	0	55
80	15	7	0	414
80	20	7	0	79
100	20	8	4 %	600
100	25	8	3 %	600

Table 2 The average RPD of the algorithms on small-sized instances

Instance			GA	SA	AIA
c	*l*	*e*			
20	5	4	0,76	1,17	0,44
	7	4	1,18	1,57	0,68
40	10	5	1,03	1,32	0,71
	15	5	1,57	2,9	0,9
60	10	6	2,37	1,16	1,14
	15	6	1,2	2,12	0,67
80	15	7	2,56	3,42	1,64
	20	7	1,12	1,59	0,78
Average			1,47	1,91	0,87

Эксперимент на крупных размерах

В этом разделе предложенные алгоритмы (GA, SA и AIA) сравниваются на наборе из 60 крупных экземпляров, упомянутых в разделе 4. Таб. 3 показаны результаты, полученные с помощью алгоритмов. На рис. 5 показаны средние интервалы RPD и наименьшей значимой разности (LSD) для трех протестированных алгоритмов. Самым эффективным алгоритмом является AIA со средним RPD 0,63%. После AIA GA дает среднее значение RPD 1,66%, тогда как алгоритм с наихудшими характеристиками - SA со средним значением RPD 2,89%.

Мы оцениваем влияние размера задачи (количество курсов) на производительность проверенных алгоритмов. Рис. 6 показывает производительность алгоритмов в зависимости от количества курсов. Что касается количества курсов, AIA сохраняет стабильную производительность в разных размерах, в то время как производительность SA ухудшается в больших размерах.

Table 3 The average RPDs obtained by the algorithms

c	*l*	Algorithms		
		GA	SA	AIA
100	20	1,32	2,18	0,49
100	30	1,2	2,24	0,68
200	30	1,45	2,59	0,51
200	50	2,12	2,71	1,01
300	50	2,07	3,76	0,41
300	70	1,83	3,91	0,66
Average		1,66	2,89	0,63

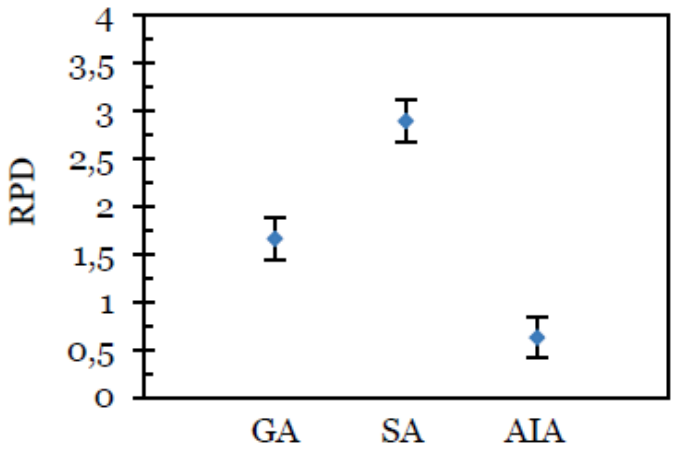


Figure 5 Means plot and LSD intervals for the different algorithms

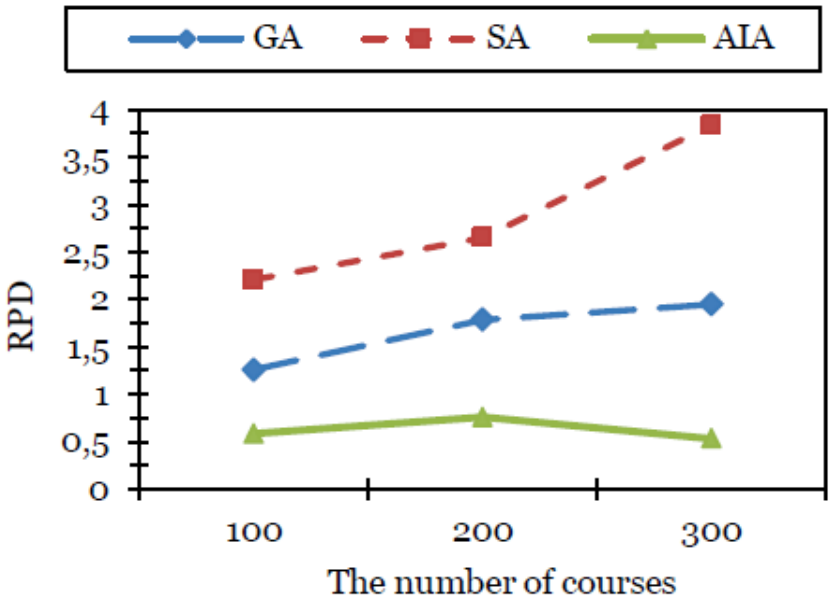


Figure 6 The average RPD of the algorithms versus the number of courses

В качестве важной проблемы мы рассматривали проблему планирования университетского курса. Целевая функция была определена для планирования курсов, чтобы максимизировать общую полезность инструктора, курсов и дней. Сначала мы смоделировали задачу математически в форме целочисленной линейной программы. Эта модель способна решать задачи до 6 курсов и до 15 инструкторов. Кроме того, мы разработали три передовые метаэвристики для решения крупногабаритных задач. Алгоритмы были основаны на алгоритмах искусственного иммунного, генетического и имитационного отжига. Они использовали новые процедуры, такие как операторы перемещения и пересечения. После настройки алгоритмов мы оценили их, сравнив с оптимальными решениями, полученными моделью. Затем алгоритмы сравнивались по ряду более крупных задач. Результаты показывают, что предложенный алгоритм искусственного иммунитета превосходит другие алгоритмы.

ГЛАВА 3. ГЕНЕТИЧЕСКИЙ АЛГОРИТМ ПЛАНИРОВАНИЯ ЗАДАЧ В СРЕДЕ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ

В настоящее время облачные вычисления широко используются в компаниях и на предприятиях. Однако при использовании облачных вычислений возникают некоторые проблемы. Основной проблемой является управление ресурсами, когда облачные вычисления предоставляют ИТ-ресурсы (например, ЦП, память, сеть, хранилище и т. Д.) На основе концепции виртуализации и принципа оплаты по мере использования. Управление этими ресурсами было предметом многих исследований. В этой статье был представлен алгоритм планирования задач, основанный на генетическом алгоритме (GA), для распределения и выполнения задач приложения. Целью этого предложенного алгоритма является минимизация времени выполнения и стоимости задач, а также максимальное использование ресурсов. Производительность этого предложенного алгоритма была оценена с использованием набора инструментов CloudSim.

Поскольку распределение облачного ресурса основано на SLA, стоимость выполнения задачи считается одним из основных параметров производительности алгоритма планирования задач [5]. С другой стороны, алгоритм планирования задач считается сложным процессом, поскольку он должен планировать большое количество задач в доступных ресурсах. С другой стороны, существует множество параметров, которые следует учитывать при разработке алгоритма планирования задач. Некоторые из этих параметров важны с точки зрения пользователя Cloud (то есть время компиляции задач, стоимость и время отклика). Другие параметры важны с точки зрения поставщика облака (т. Е. Использование ресурсов, отказоустойчивость и энергопотребление) [6].

Задача планирования задачи считается проблемой NP-Complete. Следовательно, для ее решения можно использовать подходы оптимизации с

учетом параметров производительности (то есть времени завершения, стоимости, использования ресурсов и т. Д.) [6]. Целью данной статьи является разработка алгоритма планирования задач в среде облачных вычислений на основе генетического алгоритма для распределения и выполнения независимых задач, чтобы сократить время выполнения задачи, снизить стоимость выполнения, а также максимизировать использование ресурсов.

Остальная часть статьи выглядит следующим образом: Раздел 2 обсуждает соответствующую работу. В Разделе 3 описаны принципы модифицированного планирования задач на основе GA. Конфигурация симулятора CloudSim, реализация предложенного Генетического алгоритма, а также оценка производительности обсуждаются в Разделе 4. Наконец, заключение и дальнейшая работа даны в Разделе 5.

3.1. Связанных с работой

В последние годы проблема планирования задач в распределенной среде привлекла внимание исследователей. Основной проблемой является время выполнения, которое должно быть сведено к минимуму. С другой стороны, планирование задач считается критической проблемой в среде облачных вычислений с учетом различных факторов, таких как время выполнения, общая стоимость выполнения задач всех пользователей, использование ресурса, энергопотребление и отказоустойчивость.

GE Junwei [6] представил статический генетический алгоритм с учетом общего времени выполнения задачи, среднего времени выполнения задачи и ограничения затрат.

Одной из проблем планирования является выделение правильного ресурса для прибывающих задач. Процесс динамического планирования считается сложным, если несколько задач поступают одновременно. Поэтому С. Равичандран и Д. Е. Наганатан [7] ввели систему, позволяющую избежать этой проблемы, позволяя поступившим задачам ждать в очереди, и планирование

будет пересчитываться повторно. и сортировать эти задачи. Следовательно, планирование выполняется путем извлечения первой задачи из очереди и распределения ее по ресурсу, который будет наилучшим образом соответствовать GA. Целью этой системы является максимальное использование ресурсов, а также сокращение времени выполнения.

P. Каур и C. Кингер [5] предложили расширение GA на основе алгоритма планирования задач. Они используют новую фитнес-функцию, основанную на средних и больших средних значениях. Они утверждают, что этот алгоритм может быть реализован при планировании задач и ресурсов.

Сравнительное исследование трех алгоритмов планирования задач в среде облачных вычислений - циклический перебор, приоритетный алгоритм и первые алгоритмы с кратчайшим оставшимся временем - было проведено в [8].

B. B. Кумар и C. Паланисвами [9] представили исследование, посвященное повышению эффективности алгоритма планирования задач для служб облачных вычислений в реальном времени. Кроме того, они внедрили алгоритм для использования времени выполнения, назначив высокий приоритет для задачи раннего времени завершения и меньший приоритет для проблем аборта задачи в реальном времени.

Z. Zheng и соавт. [10] предложили алгоритм, основанный на GA, для решения задачи планирования в облачной вычислительной среде, который называется Parallel Genetic Algorithm (PGA), чтобы математически добиться оптимизации или субоптимизации для задач планирования облака.

Кроме того, одной из основных целей планирования задач с точки зрения поставщика облачных услуг является максимизация прибыли за счет эффективного использования ресурсов. Поэтому K. Thyagaarajan и соавт. [11] представили модель для планирования задач в среде облачных вычислений для эффективного увеличения прибыли поставщика услуг облачных вычислений.

В [12] С. Сингх представил сложную идею о GA, представив несколько вариантов планирования задач в среде облачных вычислений. Он ввел алгоритм для решения задачи планирования задач путем изменения GA, в котором начальная популяция генерируется с помощью подхода Max-Min, чтобы получить более оптимальные результаты в терминах «рабочего времени».

3.2. Предложение генетического алгоритма планирования задач

Облачный провайдер должен гарантировать оптимальное планирование задач пользователя в среде облачных вычислений в соответствии с SLA. В то же время он должен гарантировать лучшую пропускную способность и хорошее использование облачных ресурсов.

Как правило, за счет увеличения количества пользовательских задач сложность планирования этих задач в среде облачных вычислений будет пропорционально увеличена. Поэтому облачному провайдеру нужен хороший алгоритм для планирования задач пользователей в облаке, чтобы удовлетворить QoS, минимизировать время выполнения и гарантировать хорошее использование ресурсов облака [5]. Следовательно, планирование задач классифицируется как задача оптимизации.

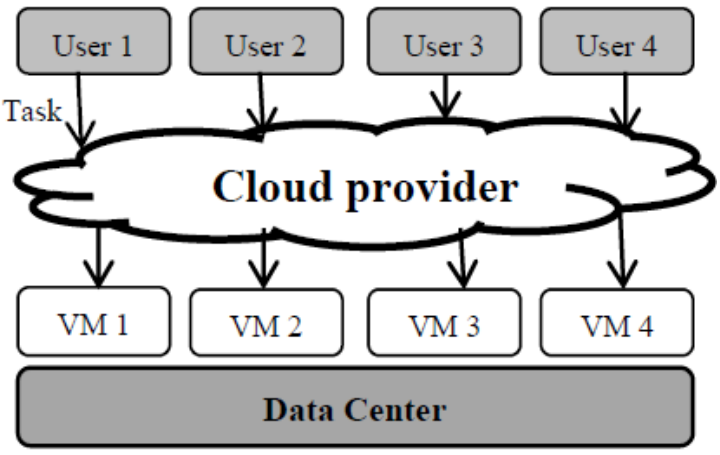


Рис. 1. Принципы планирования задач

На рис. 1 показан процесс планирования задач, в котором каждый пользователь представляет задачи своего приложения, а провайдер Cloud использует соответствующие подходы для планирования этих задач с учетом некоторых параметров оптимизации, таких как минимальный рабочий цикл, использование ресурсов и минимальная стоимость.

Следовательно, проблема оптимизации может быть решена с использованием эвристического алгоритма, такого как генетический алгоритм (GA), оптимизация роя частиц (PSO) и оптимизация колонии муравьев (ACO).

В этой работе предложенный алгоритм планирования задач в облачной среде основан на стандартном GA с некоторыми изменениями. Согласно этим модификациям, родители будут рассматриваться в каждой популяции рядом с рожденным ребенком после процесса кроссовера. Кроме того, Турнирный отбор используется для выбора лучших хромосом, чтобы преодолеть ограничение размера популяции. Поэтому предлагаемый алгоритм называется генетическим алгоритмом отбора турниров (TS-GA).

А. Генетический алгоритм

Генетический алгоритм (GA) основан на биологической концепции генерации популяции. ГА считается быстро растущей областью искусственного интеллекта [1] [2]. Дарвиновская теория эволюции была вдохновлена генетическими алгоритмами (ГА). Согласно теории Дарвина, термин «Выживание наиболее приспособленных» используется в качестве метода планирования, в котором задачи назначаются ресурсам в соответствии со значением функции пригодности для каждого параметра процесса планирования задач [13]. Основные принципы ГА описываются следующим образом [1] [2]:

1) Начальная популяция

Начальная популяция - это совокупность всех индивидуумов, которые используются в ГА, чтобы найти оптимальное решение. Каждое решение в

популяции называется индивидуальным. Каждый человек представлен в виде хромосомы для того, чтобы сделать ее пригодной для генетических операций. Из начальной популяции отбираются особи, и к ним применяются некоторые операции для формирования следующего поколения. Спаривающиеся хромосомы выбираются на основе некоторых конкретных критериев.

2) Фитнес-функция

Производительность любого человека зависит от ценности пригодности. Это мера превосходства человека в популяции. Значение пригодности показывает производительность человека в популяции. Таким образом, люди выживают или вымирают в зависимости от физической или функциональной ценности. Следовательно, фитнес-функция является мотивирующим фактором в GA.

3) Выбор

Механизм отбора используется для выбора промежуточного решения для следующего поколения на основе закона выживания Дарвина. Эта операция является направляющим каналом для GA на основе производительности. Существуют различные стратегии выбора, чтобы выбрать лучшие хромосомы, такие как колесо рулетки, стратегия Больцмана, выбор турнира и выбор на основе ранга.

4) Кроссовер

Операция кроссовера может быть достигнута путем выбора двух родительских индивидуумов, а затем создания нового отдельного дерева путем чередования и реформирования частей этих родителей. Операция гибридизации является руководящим процессом в GA и усиливает механизм поиска.

5) Мутация

После кроссовера происходит мутация. Это оператор, который вводит генетическое разнообразие в популяции. Мутация происходит всякий раз, когда популяция стремится стать однородной из-за многократного использования

операторов размножения и кроссовера. Это происходит во время эволюции в соответствии с определенной пользователем вероятностью мутации, обычно установленной на довольно низкую. Мутация изменяет одно или несколько значений генов в хромосоме от ее исходного состояния. Это может привести к совершенно новым значениям гена, добавляемым в генофонд. С этими новыми значениями генов генетический алгоритм может дать лучшее решение, чем было ранее (см. Рис. 2) [1].

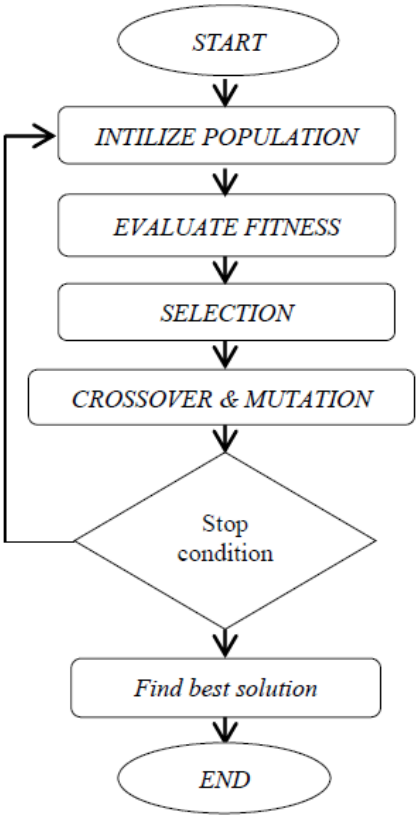


Fig. 2. Genetic Algorithm [14]

Рис. 2. Генетический алгоритм [14]

B. Предлагаемый генетический алгоритм отбора турниров (TS-GA)

В этой работе предлагается изменить ГА для решения задачи планирования задач в среде облачных вычислений, чтобы увеличить время завершения для выполнения всех задач на виртуальных машинах, в то же время минимизировать общую стоимость использования ресурса и максимизировать использование ресурса. , Основная идея этого предложенного алгоритма (то

есть TS-GA) состоит в том, что после каждого выбора в популяции существует решение, которое может удовлетворять хорошей функции пригодности, но оно не выбрано для процесса кроссовера. По предлагаемому алгоритму это решение не удаляется из совокупности, но оно выбирается и добавляется в совокупность при запуске следующей итерации. Этот шаг считается хорошим шагом, так как некоторые из итераций могут дать лучшее решение.

1) Инициализировать население

Согласно предложенному алгоритму TS-GA, популяция генерируется случайным образом с использованием закодированного двоичного кода (0, 1).

Следовательно, представление решений в планировании задач для каждого гена или (хромосомы) состоит из идентификатора виртуальной машины и идентификатора для каждой задачи, которая должна быть выполнена на этой виртуальной машине (см. Рис. 3).

Каждая виртуальная машина и выполняемые на ней задачи кодируются в двоичный бит (например, VM3: - TS4-TS8-TS9 ◊ [0110 - 0100-1000-1001]).

2) Представление Фитнес-функции

Основной целью планирования задач в облачных вычислениях является сокращение времени выполнения всех задач на доступных ресурсах. Следовательно, время выполнения задачи по as определяется с использованием уравнения «(1)» [15]:

$$\text{Completion Time} = CTmax[i,j] \quad i \in T, \ i = 1,2,3,\dots n \quad j \in VM, \ j = 1,2,3,\dots m \qquad (1)$$

Где обозначает максимальное время для завершения задачи я на. n и m обозначают количество задач и виртуальных машин соответственно.

Следовательно, чтобы уменьшить время завершения, которое можно обозначить как, время выполнения каждой задачи для каждой виртуальной машины должно быть рассчитано для целей планирования. Если скорость

обработки виртуальной машины равна, то время обработки для задачи Pi можно рассчитать по уравнению «(2)» [15]:

$$Pij = \frac{Ci}{Psj} \qquad (2)$$

Где время обработки задачи Pi на VMj и Ci вычислительная сложность задачи Pi

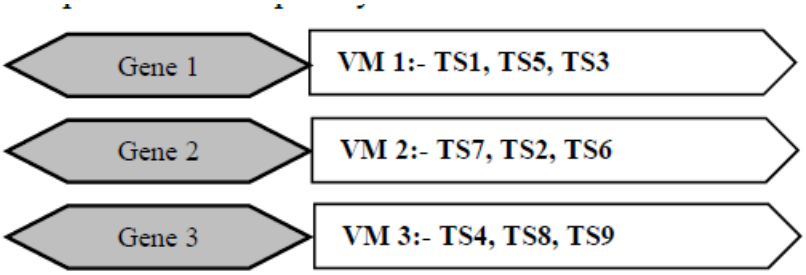


Рис. 3. Представление задач и ВМ

Время обработки каждой задачи в виртуальной машине можно рассчитать по уравнению «(3)» [15]:

$$Pj = \sum_{i=1}^{n} Pij \qquad (3)$$

3) Процесс выбора

Выбор турниров в вычислительном отношении более эффективен и более поддается параллельной реализации [16]. Поэтому разработанный алгоритм TS-GA Tournament Selection используется для преодоления ограничения численности населения. Два человека выбираются случайным образом из населения. Случайное число r затем выбирается между 0 и 1. Если r <k (где k является

параметр, например, 0,75), установщик двух индивидуумов выбирается в качестве родителя; в противном случае выбирается менее подходящий человек.

Невыбранные лица затем возвращаются к исходному населению и могут быть выбраны снова.

4) Кроссовер

В предлагаемом алгоритме TS-GA новый кроссовер использовался не так, как в оригинальном GA. Следовательно, две хромосомы, которые выбраны для кроссоверного процесса с целью получения двух потомков, также будут считаться потомками. Итак, предлагаемый кроссовер производит четверых детей (см. Рисунок 4). После этого из них выбираются два лучших ребенка.

5) Инициализация субпопуляции

После каждой итерации субпопуляции (то есть новые популяции после кроссовера) добавляются в старые популяции (то есть, родители). Этот шаг может увеличить разнообразие населения. Эта идея представлена модифицированным алгоритмом TS-GA.

6) Держите лучшее решение

Существует решение, которое может удовлетворять хорошей фитнес-функции, но оно не выбрано в процессе кроссовера. По предлагаемому алгоритму TS-GA это решение не удаляется из совокупности, но оно выбирается и добавляется в совокупность при запуске следующей итерации. Этот шаг считается хорошим шагом, так как некоторые из итераций могут дать лучшее решение.

Как правило, согласно модифицированному алгоритму TS-GA, был представлен ряд модификаций. Эти модификации заключаются в следующем.

- Турнир используется вместо колеса рулетки в процессе выбора, чтобы выбрать лучшее решение.
- Решения, не выбранные в процессе отбора, рассматриваются и добавляются к новому населению. Это может помочь в создании лучшего решения в следующих поколениях.

• Новый кроссовер вводится с учетом индивидуумов родителей как нового ребенка (см. Рис. 4)

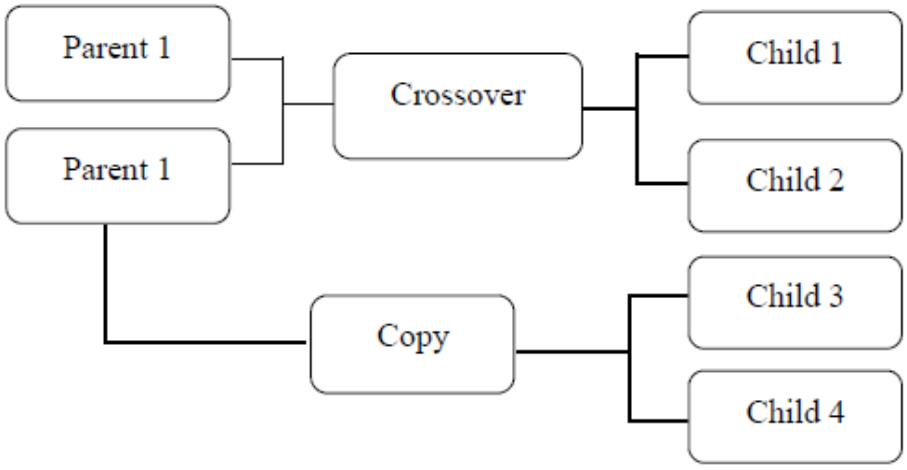


Рис. 4. Процесс кроссовера

После каждой итерации субпопуляции (то есть новые популяции после кроссовера) добавляются в старые популяции (то есть, родители).

3.3. По оценка эффективности

В этом разделе представлена экспериментальная оценка предложенного алгоритма TS-GA на исходных алгоритмах GA и Round-Robin, начиная с описания среды эксперимента.

Экспериментальная среда

Инструментарий CloudSim помогает исследователям моделировать среду облачных вычислений, где он выпущен Лабораторией облачных вычислений и распределенных систем, Мельбурнский университет [17]. Другими словами, он предоставляет возможности моделирования и симуляции среды облачных вычислений. По словам CloudSim, пользователь пытается подать свои запросы в виде облачков. Каждое облачко имеет свойства размера файла, количества выполняемых команд и т. Д. Эти кластеры будут отправлены брокеру для планирования на виртуальные машины в соответствии с расписанием. CloudSim имеет преимущество в построении политик, управляемых брокерами. Определенный класс в CloudSim, VM, представляет виртуальную машину,

которую можно создать на хостах. Создание хостов зависит от брокера, в котором он выделяет каждую виртуальную машину другому хосту. Центр обработки данных имеет возможность хранить максимальное количество хостов, и брокер может динамически изменять настройки хостов и виртуальных машин (см. Рис. 5) [17].

3.4. Результаты эксперимента

Используя инструментарий CloudSim, предложенный TS-GA реализован, и было проведено сравнительное исследование среди трех алгоритмов; Round-Robin (RR), GA по умолчанию и улучшенные алгоритмы TS-GA. Пять параметров рассматриваются для оценки производительности. Эти параметры - время завершения, стоимость, использование ресурсов, ускорение и эффективность.

Время завершения:

Таблица 1 и рис. (6) представляют время завершения RR, GA по умолчанию и предлагаемых алгоритмов TS-GA с использованием 8 ВМ.

Таблица 1. Время завершения алгоритмов RR, GA и TS-GA с использованием восьми VMS

No. of Task	RR	GA	TS-GA	Improve TS-GA vs. GA	No. VM
25	174.78	292.51	122.19	58.22 %	8
50	578.77	420.21	286.12	31.91 %	
75	582.16	680.84	405.97	40.37 %	
100	954.37	812.89	513.7	36.8 %	

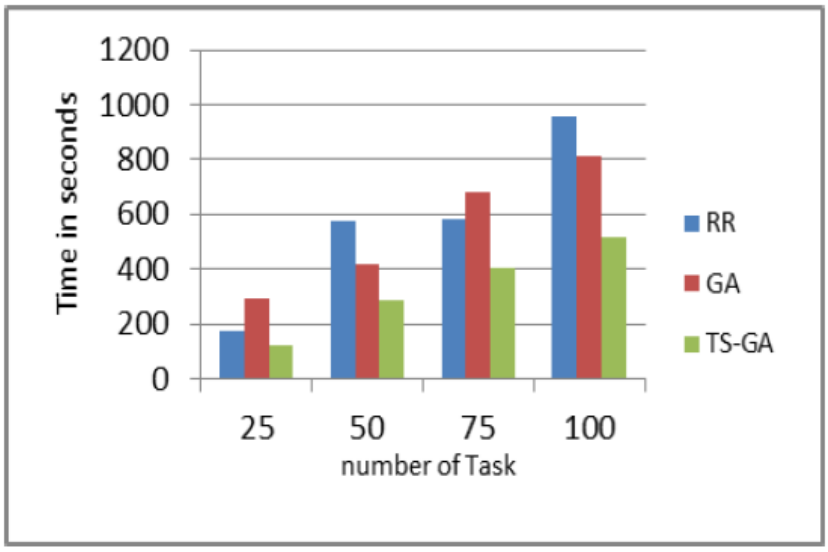


Рис. 6. Время завершения сравнения трех алгоритмов RR, GA и TS-GA

Согласно результатам на рис. (6), установлено, что время завершения предложенного алгоритма TS-GA сокращается на (41,83%) и (39,26%) для алгоритмов GA по умолчанию и RR соответственно.

2) Стоимость исполнения:

Кроме того, общая стоимость выполнения всех задач на доступных виртуальных машинах рассчитывается как «(4)» [18]:

$$Total\ Cost = \frac{Task\ length\ *\ Cost\ per\ seconds}{VM\ mips} + Processing\ Cos \quad (4)$$

Таблица 2. и рис. (7) представляют стоимость выполнения RR, GA по умолчанию и предложенных алгоритмов TS-GA с использованием 8 виртуальных машин.

ТАБЛИЦА II. ИСПОЛНИТЕЛЬНАЯ СТОИМОСТЬ АЛГОРИТМОВ RR, GA И TS-GA

No. of Task	RR	GA	TS-GA	IMPROVE TS-GA vs. GA	No. VM
25	3017.96	3012.98	2915.21	3.24 %	8
50	7543.22	6813.46	6603.88	1.3 %	
75	10517.14	10549.22	9910.48	6.05 %	
100	13515.05	13118.87	12618.22	3.81 %	

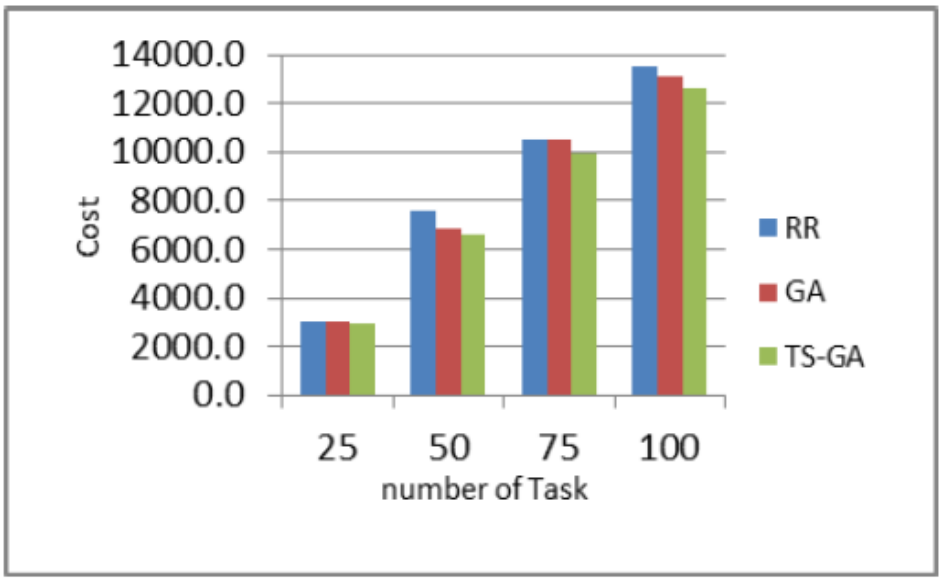


Рис. 7. Сравнение стоимости трех алгоритмов RR, GA и TS-GA

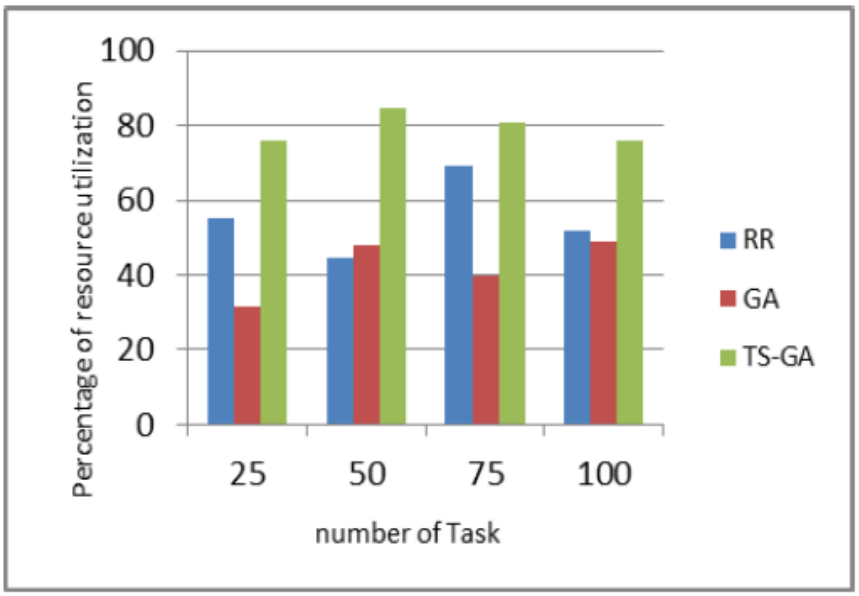
Согласно результатам на рис. (7), стоимость предложенного алгоритма TS-GA снижается на (3,6%) и (6,07%) по сравнению с алгоритмами GA и RR по умолчанию соответственно.

3) Использование ресурсов:

С другой стороны, использование ресурсов представляет собой соотношение между общим временем занятости виртуальной машины и общим временем завершения параллельного приложения. Он определяется как «(5)» [19]:

$$\text{utilization} = \frac{\text{final VMs available time}}{\#\text{VMs} \ * \ \text{schedule time}} * 100 \qquad (5)$$

No. of Task	RR	GA	TS-GA	IMPROVE TS-GA vs. GA	No. VM
25	55.13	31.51	75.89	58.47	8
50	44.36	47.88	84.43	43.29	
75	69.4	39.96	80.75	50.51	
100	52.04	48.7	75.78	35.73	



Согласно результатам на рис. (8), установлено, что использование ресурсов предложенного алгоритма TS-GA улучшено на (47%) и (30,04%) относительно алгоритмов GA и RR по умолчанию соответственно.

4) Ускорение и эффективность:

Ускорение и эффективность для каждой виртуальной машины рассчитываются как «(6)», «(7)» следует [19]:

$$\text{Speedup} = \frac{final\ sum\ of\ excute\ time\ of\ tasks\ to\ one\ VM/\#VMs}{schedule\ time} \qquad (6)$$

$$\text{Efficiency} = \frac{speedup}{\#\ VMs} \qquad (7)$$

No. TASK	Algorithm	Speedup	Efficiency
25	RR	0.193	0.241
	GA	0.143	0.179
	TS-GA	0.248	0.311
50	RR	0.314	0.393
	GA	0.419	0.524
	TS-GA	0.558	0.697
75	RR	0.598	0.748
	GA	0.543	0.679
	TS-GA	0.829	1.037
100	RR	0.622	0.778
	GA	0.691	0.864
	TS-GA	1.105	1.319

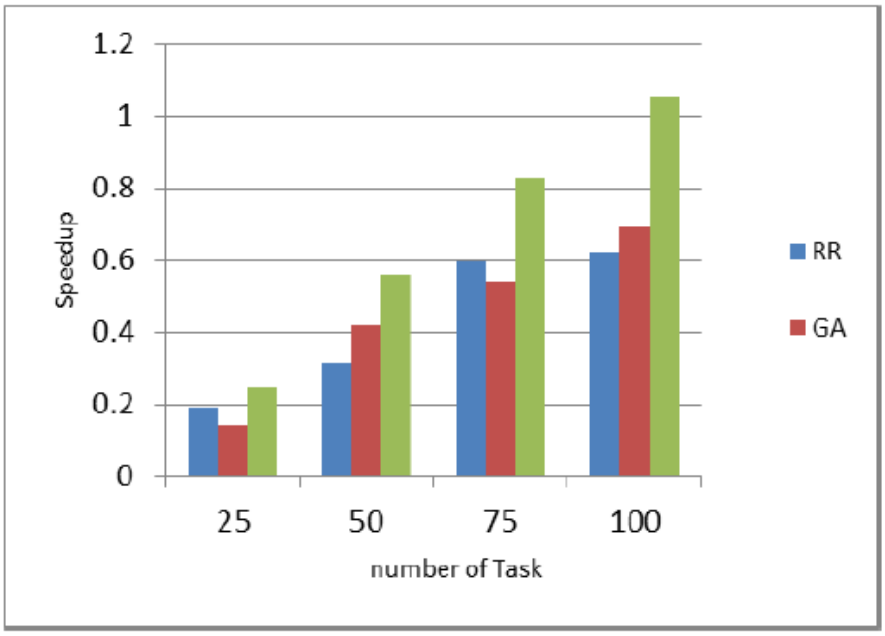


Рис. 9. Сравнение ускорения трех алгоритмов RR, GA и TS-GA

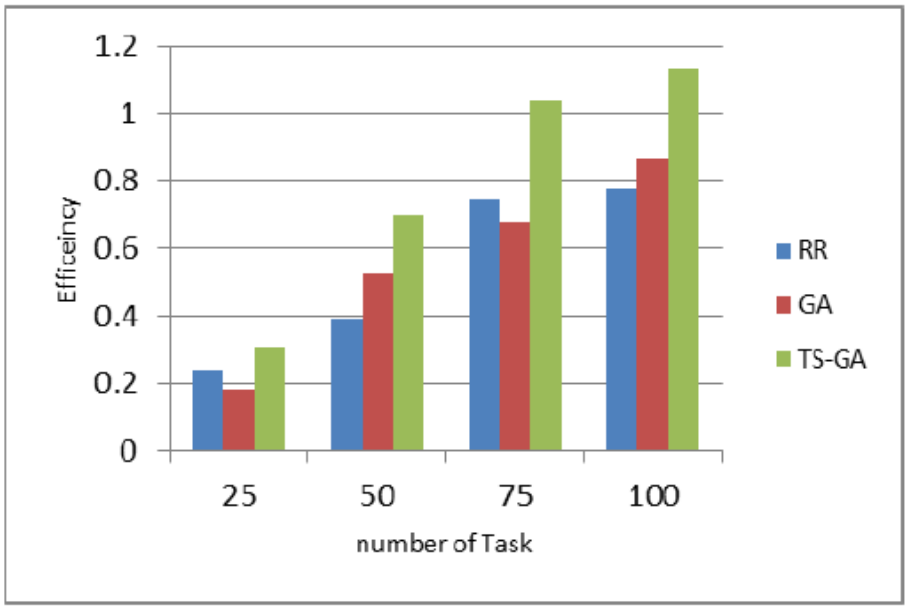


Рис. 10. Сравнение эффективности трех алгоритмов RR, GA и TS-GA

Согласно результатам на рис. (9), установлено, что ускорение предлагаемого алгоритма TS-GA улучшено на (34,03%) и (33,65%) по сравнению с алгоритмами GA и RR по умолчанию соответственно. Кроме того, эффективность предложенного алгоритма TS-GA повышается на (34,06%) и (33,66%) по сравнению с алгоритмами GA и RR по умолчанию соответственно (см. Рис. 10).

Среднее улучшение ускорения и эффективности предложенного алгоритма TS-GA относительно алгоритмов GA и RR по умолчанию представлено в таблице 5 и таблице 6 соответственно.

ТАБЛИЦА V. СРЕДНЕГО УЛУЧШЕНИЯ УСКОРЕНИЯ И ЭФФЕКТИВНОСТИ ДЛЯ АЛГОРИТМА TS-GA, ОТНОСЯЩЕГОСЯ К АЛГОРИТМУ GA ПО УМОЛЧАНИЮ

No. TASK	**25**	**50**	**75**	**100**	**Average improvement**
Speedup	42.35	24.82	34.51	34.45	34.03 %
Efficiency	42.44	24.82	34.52	34.49	34.06 %

No. TASK	25	50	75	100	Average improvement
Speedup	22.22	43.57	27.86	40.98	33.65 %
Efficiency	22.23	43.58	27.88	40.9	33.69 %

ЗАКЛЮЧЕНИЕ И БУДУЩАЯ РАБОТА

В этой статье предлагается улучшенный генетический алгоритм для задачи планирования задач в среде облачных вычислений. Предложенный алгоритм направлен на минимизацию времени завершения и затрат, а также на максимальное использование ресурсов. Время завершения для предложенного алгоритма TS-GA сокращается на (41,83%) и (39,26%) об алгоритмах GA и RR по умолчанию, соответственно. Стоимость предложенного алгоритма TS-GA снижается на (3,6%) и (6,07%) по сравнению с алгоритмами GA и RR по умолчанию соответственно. Использование ресурсов предложенного алгоритма TS-GA улучшено на (47%) и (30,04%) для алгоритмов GA и RR по умолчанию, соответственно. Ускорение предлагаемого алгоритма TS-GA улучшено на (34,03%) и (33,65%) по сравнению с алгоритмами GA и RR по умолчанию, соответственно. Эффективность предложенного алгоритма TS-GA повышена на (34,06%) и (33,66%) по сравнению с алгоритмами GA и RR по умолчанию соответственно.

Для дальнейшей работы предложенный алгоритм может быть расширен, чтобы добавить возможность динамической характеристики BM через прогон GA. Кроме того, можно добавить больше параметров в зависимости от требований пользователей.

ЛИТЕРАТУРА

[1] S. H. Jang, T. Y. Kim, J. K. Kim, and J. S. Lee, "The study of genetic algorithm-based task scheduling for cloud computing," International Journal of Control and Automation, vol. 5, pp. 157-162, 2012.

[2] T. Goyal and A. Agrawal, "Host Scheduling Algorithm Using Genetic Algorithm In Cloud Computing Environment," International Journal of Research in Engineering & Technology (IJRET) Vol, vol. 1, 2013.

[3] B. Furht, "Armando Escalante Handbook of Cloud Computing," ISBN 978-1-4419-6523-3, Springer2010.

[4] F. Etro, "Introducing Cloud Computing," in London Conference on Cloud Computing For the Public Sector, 2010, pp. 01-20.

[5] R. Kaur and S. Kinger, "Enhanced Genetic Algorithm based Task Scheduling in Cloud Computing," International Journal of Computer Applications, vol. 101, 2014.

[6] J. W. Ge and Y. S. Yuan, "Research of cloud computing task scheduling algorithm based on improved genetic algorithm," in Applied Mechanics and Materials, 2013, pp. 2426-2429.

[7] S. Ravichandran and D. E. Naganathan, "Dynamic Scheduling of Data Using Genetic Algorithm in Cloud Computing," International Journal of Computing Algorithm, vol. 2, pp. 127-133, 2013.

[8] V. Vignesh, K. Sendhil Kumar, and N. Jaisankar, "Resource management and scheduling in cloud environment," International Journal of Scientific and Research Publications, vol. 3, p. 1, 2013.

[9] V. V. Kumar and S. Palaniswami, "A Dynamic Resource Allocation Method for Parallel DataProcessing in Cloud Computing," Journal of computer science, vol. 8, p. 780, 2012.

[10] Z. Zheng, R. Wang, H. Zhong, and X. Zhang, "An approach for cloud resource scheduling based on Parallel Genetic Algorithm," in Computer Research and Development (ICCRD), 2011 3rd International Conference on, 2011, pp. 444-447.

[11] K. Thyagarajan, S. Vasu, and S. S. Harsha, "A Model for an Optimal Approach for Job Scheduling in Cloud Computing," in International Journal of Engineering Research and Technology, 2013.

[12] S. Singh and M. Kalra, "Scheduling of Independent Tasks in Cloud Computing Using Modified Genetic Algorithm," in Computational Intelligence and Communication Networks (CICN), 2014 International Conference on, 2014, pp. 565-569.

[13] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities," in High Performance Computing & Simulation, 2009. HPCS'09. International Conference on, 2009, pp. 1-11.

[14] J. S. Raj and R. M. Thomas, "Genetic based scheduling in grid systems: A survey," in Computer Communication and Informatics (ICCCI), 2013 International Conference on, 2013, pp. 1-4.

[15] B. Kruekaew and W. Kimpan, "Virtual Machine Scheduling Management on Cloud Computing Using Artificial Bee Colony," in Proceedings of the International MultiConference of Engineers and Computer Scientists, 2014.

[16] M. Mitchell, An introduction to genetic algorithms: MIT press, 1998.

[17] R. N. Calheiros, R. Ranjan, C. A. De Rose, and R. Buyya, "Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services," arXiv preprint arXiv:0903.2525, 2009.

[18] R. Sahal and F. A. Omara, "Effective virtual machine configuration for cloud environment," in Informatics and Systems (INFOS), 2014 9th International Conference on, 2014, pp. PDC-15-PDC-20.

[19] D. M.Abdelkader, F.Omara," Dynamic task scheduling algorithm with load balancing for heterogeneous computing," system Egyptian Informatics Journal, Vol.13, PP.135–145, 2012.