Application of the modified ant colony algorithm for solving the problem of scheduling of distributed enterprises

Danil D. Bukhovtsev

Siberian Federal University, Krasnoyarsk, Russian Federation

E-mail: wav9@yandex.ru

Abstract. The task of scheduling of distributed enterprises is to assign tasks to geographically remote enterprises and to determine a convenient work schedule for each of the enterprises. The goal of solving this problem is to minimize the total production time at all enterprises. This article is the first step towards solving the scheduling problem using different versions of the ant colony algorithm: the classic ant colony algorithm, the ant colony system algorithm, and the modified ant colony algorithm.

Keywords: ant algorithm, scheduling, modification

УДК 004.021

Применение модифицированного алгоритма муравьиной колонии для решения задачи календарного планирования распределенных предприятий

Данил Денисович Буховцев

Сибирский федеральный университет, Красноярск, Российская Федерация

E-mail: wav9@yandex.ru

Аннотация. Задача календарного планирования распределенных предприятий заключается в назначении заданий территориально удаленным предприятиям и определении удобного рабочего графика для каждого из предприятий. Целью решения данной задачи является минимизация общего времени изготовления на всех предприятиях. Данная статья является первым шагом к решению задачи календарного планирования с использованием различных версий алгоритма муравьиной колонии: классический муравьиный алгоритм, алгоритм системы муравьиных колоний и модифицированный алгоритм муравьиной колонии.

Ключевые слова: муравьиный алгоритм, календарное планирование, модификация.

1. Введение

Производственная отрасль сталкивается с множеством проблем: обеспечение роста прибыли, повышение производительности и снижение затрат при быстром реагировании на требования клиентов. Для сохранения конкурентоспособности и приближения к рыночным требованиям, промышленные компании всё чаще склоняются к распределению. В связи с этим структура их цехов изменяется от классических к распределенным.

В этой статье мы обратимся к задаче календарного планирования распределенных предприятий, которую можно рассматривать как расширение простой задачи календарного планирования предприятия. Планирование заданий представляет собой некоторый набор f идентичных по структуре цехов, которые географически распределены по разным районам. В каждом цехе имеется m станков, которые могут обрабатывать определенное количество заданий. Очевидно, что задача календарного планирования для распределенных предприятий намного сложнее, чем задача классического планирования, поскольку необходимо решить две проблемы: распределить задания по подходящим цехам и упорядочить операции на станках с

целью минимизации одного или нескольких заранее определенных критериев производительности.

Решением рассматриваемой проблемы является минимизация максимальной продолжительности обработки заданий $C_{\max}$ среди всех распределенных цехов, использующих три версии алгоритма муравьиной колонии.

Оставшаяся часть статьи организована следующим образом. В разделе 2 приводится небольшой литературный обзор существующих работ по данной теме, оговариваются некоторые нюансы распределенного календарного планирования. В разделе 3 описывается популярная модель представления календарного планирования – дизъюнктивный граф. В разделе 4 описан эффективный способ распределения заданий по цехам. В разделе 5 рассматриваются 3 версии алгоритма муравьиной колонии. В разделе 6 проводится оценка эффективности алгоритмов. В разделе 7 подводятся основные итоги.

2. Современное состояние проблемы распределенного планирования

За последние три десятилетия задачи планирования стали пользоваться особой популярностью у исследователей и промышленников. В связи с тенденцией к глобализации календарное планирование превратилось из классического в распределенное, требующее к себе особого внимания. В литературе лишь небольшое количество исследователей обращалось к данной проблеме, а используемые ими методы для её решения весьма ограничены. Задачу распределенного календарного планирования можно сформулировать следующим образом: имеется множество независимых заданий $J = \{j_1, \dots, j_n\}$, где каждое задание состоит из упорядоченного набора операций. Каждая операция должна выполняться на определенном станке m из набора $M = \{i_1, \dots, i_m\}$ географически распределённых на f цехах, каждый из которых обладает одинаковым набором M. Группой исследователей во главе с Х. Джиа был предложен подход на основе генетического алгоритма, чтобы облегчить взаимодействие между географически распределенными предприятиями [1]. В другой статье этих же исследователей, для решения аналогичной проблемы был представлен модифицированный генетический алгоритм, в котором использовался двухэтапный метод кодирования для воздействия на задания и операции [2]. Позже они усовершенствовали предложенный алгоритм и интегрировали его с диаграммой Ганта, чтобы найти оптимальное рабочее время для каждого из предприятий [3].

В [4] дан подробный литературный обзор задачи календарного планирования распределенных предприятий и описана классификация используемых методов. Основная цель календарного планирования – составить оптимальное расписание, которое смогло бы минимизировать заданный критерий, связанный, в большинстве случаев, со временем.

Существуют различные ограничения, связанные как с заданиями, так и со станками. Распределенное календарное планирование предполагает следующий ряд допущений:

- Все задания независимы и доступны для выполнения в момент времени $t = 0$.
- Все станки постоянно доступны.
- После того, как задание назначено конкретному цеху, его нельзя передать другому, так как все операции должны выполняться в одном цеху.
- Все задания могут быть выполнены в любых цехах.
- Нет никаких ограничений по приоритету между операциями различных заданий.
- Каждая операция должна быть обработана в течение непрерывного фиксированного периода времени на некотором станке.
- Одно задание может быть выполнено на одном станке, а один станок может выполнять одно задание одновременно.
- Одна операция не может выполняться на одном станке дважды.
- Время, предназначенное для наладки станков, и время перехода между операциями ничтожно малы.

3. Представление задачи распределенного планирования

Дизъюнктивный граф является одной из наиболее актуальных моделей, используемых для описания задач планирования [5]. Ниже приводится обобщение такой модели. Для каждого цеха используется дизъюнктивный граф $D = (N, A, E)$, где:

- N – набор вершин, соответствующих всем операциям O, выполняемым в цехе;
- A – набор конъюнктивных направленных дуг, основанных на правилах приоритета;
- E – набор дизъюнктивных неориентированных ребер, которые соединяют две операции из двух разных заданий, выполняемых на одном станке и в одном цехе.

На дизъюнктивном графе планирования N, A, и E определяются следующим образом:

- $N = O \cup \{u_0\} \cup \{u_{N+1}\}$, где $\{u_0\}$ и $\{u_{N+1}\}$ – фиктивные вершины, определяющие начало и конец работ в цехе;
- $A = \{u_{ij}, u_{ij+1}\}, u_{ij} \to u_{ij+1}$ – последовательность операций для задания, $J_i \cup \{u_0, u_{1j}\} : u_{1j}$ – первая операция в задании, $J_i \cup \{u_{im}, u_{N+1}\} : u_{1m}$ – последняя операция задания J_i;
- $E = \{u_{ij}, u_{hj}\}$.

Вес ребра связаны со временем обработки p_{ij}, при этом все операции $u_{ij} \in O, u_0$ и u_{N+1} имеют нулевой вес. Матрица времени обработки представлена в таблице 1.

Таблица 1. Матрица времени обработки.

Задание	Станок		Маршрут обработки
	1	2	
1	4	7	{1, 2}
2	6	6	{2, 1}
3	5	6	{2, 1}
4	6	4	{1, 2}
5	3	5	{2, 1}
6	4	4	{2, 1}

На рисунке 1 показано графическое представление распределенного календарного планирования в виде дизъюнктивного графа. После распределения заданий по цехам с использованием правила назначения, можно увидеть, что задания 1, 3 и 5 попали в 1 цех, а задания 2, 4 и 5 – во 2 цех. Кортежи типа «(задание, станок) время», расположенные около вершин с номерами операций, обозначают номер задания, номер станка и время обработки операции соответственно. Например, (3, 1) 5 рядом с вершиной 3 в 1 цехе обозначает, что операция 1 принадлежит заданию 3 и будет выполняться на станке 1 с временем обработки 5. На рисунке 2 приводится сравнение продолжительности обработки между 1 и 2 цехами. Продолжительность обработки в 1 цехе составляет 17, а во 2 цехе – 15, что позволяет сделать вывод, что максимальная продолжительность обработки равна наибольшей продолжительности обработки между двумя цехами, то есть 17.

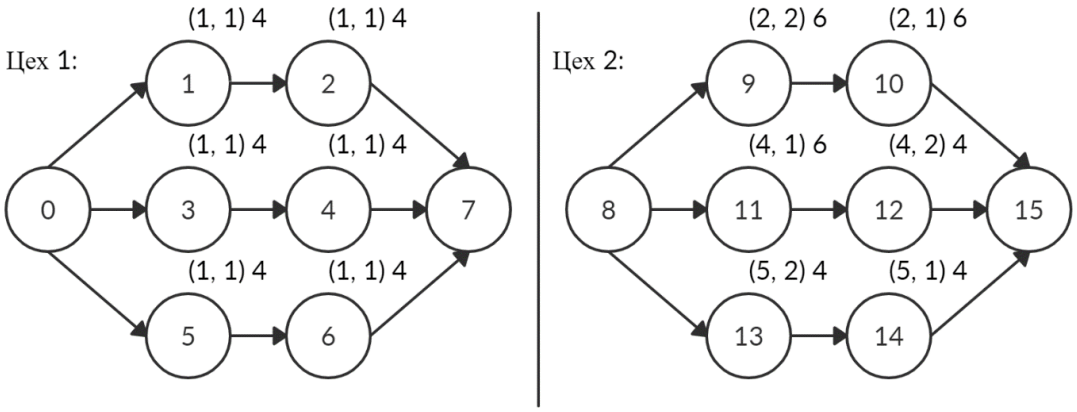

Рисунок 1. Дизъюнктивный граф представления календарного планирования с f = 2, n = 6 и m = 2.

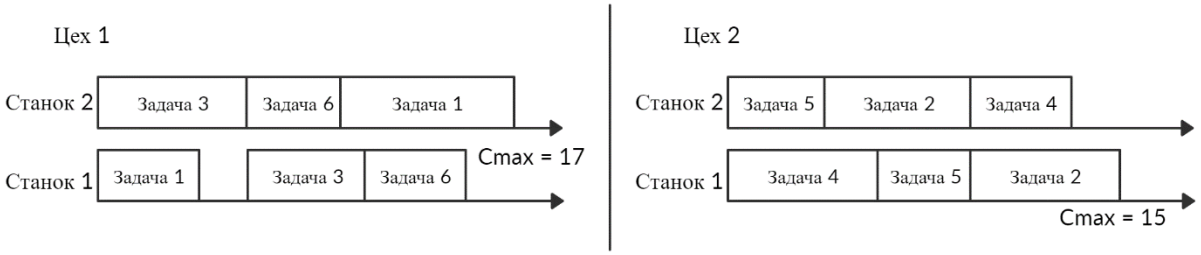

Рисунок 2. Представление календарного планирования в виде диаграммы Ганта с $f = 2$, $n = 6$ и $m = 2$.

4. Этап распределения заданий по цехам

Основным этапом в решении задачи календарного планирования является назначение заданий соответствующим цехам, целью которого является распределение нагрузки. В нашем случае мы будем использовать правило назначения, введенное в [6]. С помощью данного правила можно рассчитать рабочую нагрузку для каждого станка i, обрабатывающего задание j:

$$W(j,i) = \sum_{k \in R_{j,i}} p_{j,k} + p_{j,i}, \forall\, i,j \qquad (1)$$

где $R_{j,i}$ – набор всех станков, предшествующих станку i при обработке задания j;

$p_{j,i}$ – время обработки задания j на станке i.

После расчета рабочей нагрузки задания ранжируются в порядке убывания – от наибольшей нагрузки к наименьшей. Предположим, что имеются f цехов. Первые n заданий назначены $\{1, \dots, n\}$ цехам соответственно. Рабочая нагрузка станков в цехах становится равной нагрузке назначенных заданий. Для цехов f определяется максимальная рабочая нагрузка. Чтобы назначить следующее задание, максимальная нагрузка рассчитывается заново, но с учетом уже распределенных заданий. Затем, задание передается цеху с минимальной рабочей нагрузкой. Алгоритм повторяется до тех пор, пока не будут распределены все задания. Чтобы лучше понять данную концепцию, рассмотрим предыдущий пример с 2 цехами, 6 заданиями и 2 станками (таблица 1). Рабочая нагрузка каждой операции приведена в таблице 2.

Таблица 2. Рабочая нагрузка станков.

Задание	Станок		Общая нагрузка	Ранг
	1	2		
1	4	11	15	4
2	12	6	18	1
3	11	6	17	2
4	6	10	16	3
5	8	5	13	5
6	8	4	12	6

Отсортируем задания по убыванию общей нагрузки – 2, 3, 4, 1, 5, 6. Задание 2 и задание 3 назначены на цехи 1 и 2 соответственно. Нагрузка станков 1 и 2 в первом цехе составляет 12 и 6 соответственно. Нагрузка станков 1 и 2 во втором цехе составляет 11 и 6 соответственно. Таким образом, максимальная нагрузка цехов составляет 12 и 11 соответственно. Чтобы назначить следующее задание (задание 4), рассчитывается максимальная рабочая нагрузка. Если задание 4 будет назначено цеху 1, рабочие нагрузки станут 18 и 16, а если оно будет назначено цеху 2 – 17 и 16. Максимальная загруженность цехов 1 и 2 – 18 и 17 соответственно. Следовательно, задание 4 назначается цеху 2. Данные шаги повторяются для всех оставшихся заданий.

Данный метод эффективен для сбалансированного распределения рабочих нагрузок на различных предприятиях.

5. Предлагаемые алгоритмы

5.1. Классический муравьиный алгоритм и алгоритм системы муравьиных колоний

После этапа распределения заданий между соответствующими цехами следует этап их упорядочивания. Для этого будут применяться три версии алгоритма муравьиной колонии: классический муравьиный алгоритм, алгоритм системы муравьиных колоний и модифицированный алгоритм муравьиной колонии. Первый вариант муравьиного алгоритма был предложен доктором философии М. Дориго [7]. Основная идея данного алгоритма – имитировать поведение реальных муравьев для решения задач оптимизации [8]. В природе муравьи способны найти кратчайший путь между источником пищи и своим гнездом благодаря их коллективному поведению. Во время движения они оставляют на земле специальный след, называемый феромоном, который привлекает других муравьев колонии.

Основные этапы рассматриваемых алгоритмов представлены ниже:

Шаг 1: в начале алгоритма инициализируются все параметры, включая начальное значение феромона на ребрах.

Шаг 2: муравьи размещаются в стартовых вершинах. В нашем случае муравьи размещаются на первой операции каждого задания.

Шаг 3: с помощью правила перехода находим вероятность перехода муравьев между соответствующими вершинами, где (2) применяется для классического муравьиного алгоритма, а (3) – для алгоритма системы муравьиных колоний:

$$P(i,s)(t) = \begin{cases} \dfrac{[\tau_{i,s}(t)]^{\alpha} \times \left[\frac{1}{d_{i,s}}\right]^{\beta}}{\sum_{j \epsilon V}[\tau_{i,j}(t)]^{\alpha} \times \left[\frac{1}{d_{i,j}}\right]^{\beta}}, \\ 0, \text{в противном случае} \end{cases} \quad (2)$$

$$s = \begin{cases} argmax_{u \epsilon V}\left[\tau_{i,j}(t)\right]^{\alpha} \times \left[\frac{1}{d_{i,j}}\right]^{\beta}, \text{если } q \leq q_0, \\ S, \text{в противном случае} \end{cases} \quad (3)$$

где $\tau_{i,j}$ – количество феромона между вершинами i и j;

$d_{i,j}$ – эвристическое расстояние между вершинами i и j. В нашем случае $d_{i,j}$ – это время обработки операции;

q – случайное число, равномерно распределенное в отрезке [0; 1];

q_0 – параметр, задаваемый пользователем, находящийся в отрезке [0; 1];

$p_{i,j}$ – вероятность перехода из вершины i в вершину j;

α, β – параметры, регулирующие влияние количества феромона в зависимости от эвристического расстояния;

S – параметр, выбираемый случайным образом, в соответствии с (2).

Как уже было отмечено ранее, важным отличием между данными алгоритмами является форма решающего правила, используемого муравьями в процессе перехода между вершинами. В алгоритме системы муравьиных колоний вероятность перехода от вершины i к вершине j зависит от случайной величины q, равномерно распределенной в отрезке [0; 1], и параметра q_0. Если $q \leq q_0$, то из возможных вариантов выбирается тот, который максимизирует произведение $\left[\tau_{i,j}(t)\right]^{\alpha} \times \left[\frac{1}{d_{i,j}}\right]^{\beta}$. В противном случае используется то же уравнение, что и для классического муравьиного алгоритма.

Шаг 4: Локальное обновление феромона (применимо только для алгоритма системы муравьиных колоний). В процессе построения маршрута муравей будет изменять количество феромона на посещенных ребрах, применяя правило локального обновления:

$$\tau_{i,s}(t+n) = (1-\rho) \times \tau_{i,s}(t) + \rho \times \tau_0(t+n), \quad (4)$$

где ρ – коэффициент испарения феромона, находящийся в интервале (0; 1).

Задача правила локального обновления заключается в том, чтобы сделать посещаемые ребра менее привлекательными для муравьев, косвенно способствуя исследованию еще не посещенных ребер.

Шаг 5: Глобальное обновление феромона (5), состоящее из двух основных этапов:

- Фаза испарения, при которой часть феромона испаряется, чтобы разнообразить процедуру поиска и дать возможность найти другие решения;
- Фаза подкрепления, при которой каждый муравей оставляет некоторое количество феромона пропорционально сгенерированным решениям.

$$\tau_{i,s}(t+n) = (1-\varepsilon) \times \tau_{i,s}(t) + \varepsilon \times \tau_{i,s}(t+n), \quad (5)$$

$$\Delta\tau_{i,s}(t+n) = \frac{Q}{C_{\max}}, \quad (6)$$

где ε – параметр затирания феромона, находящийся в интервале (0; 1);

Q – константное значение.

Глобальное обновление феромона применяется ко всем решениям, найденным классическим муравьиным алгоритмом, в то время как в алгоритме системы муравьиных колоний оно применяется только к лучшему найденному решению. Алгоритм повторяется до тех пор, пока не будет достигнуто условие завершения, которое в нашем случае является фиксированным числом итераций.

Итак, после того, как все задания распределяются между цехами с использованием правила назначения, происходит упорядочивание с применением алгоритма муравьиной колонии. В каждом цехе имеется определенное количество заданий, на которые назначаются муравьи. Каждый муравей построит свое собственное законченное решение, выполнив основные шаги алгоритма, описанные выше. В конце алгоритма у нас будет набор решений, из которых мы выберем наилучшее, имеющее минимальную продолжительность обработки среди всех цехов.

5.2. Модифицированный алгоритм муравьиной колонии

В данном разделе представлен модифицированный алгоритм муравьиной колонии. Основная задача алгоритма – улучшить решение, получаемое при помощи алгоритма системы муравьиных колоний, путем применения процедуры локального поиска.

Рассмотренные ранее алгоритмы имеют ряд недостатков из-за рандомизированного характера, который позволяет принимать вероятностные решения при построении решения. На больших данных данные алгоритмы не всегда генерируют оптимальное решение проблемы.

Предлагаемая модификация призвана повысить качество решений. Полученное алгоритмом системы муравьиных колоний решение используется в качестве начального. Прежде всего, на всех станках определяется последнее запланированное задание. Затем, происходит поиск всех неактивных интервалов времени станков и вставка операции выбранного задания в эти интервалы. Чтобы поместить операцию в неактивный интервал, необходимо проверить следующие ограничения:

- Время окончания выбранной операции > Время окончания неактивного интервала;
- Время обработки выбранной операции ≤ Длина интервала.

В представленном ниже псевдокоде отражены основные этапы предлагаемого алгоритма:

Листинг 1. Псевдокод модифицированного алгоритма.

```
Begin
Инициализация параметров, установка начального значения феромона
While Критерий остановки не выполнен do
Размещение муравьев в начальных вершинах
Repeat
For количество муравьев do
Выбор следующей вершины согласно правилу перехода
Локальное обновление феромона
End for
Until каждый муравей построил решение
Обновление лучшего найденного решения
Глобальное обновление феромона
End While
For количество найденных решений
Определение неактивных временных интервалов станков
if (Время окончания выбранной операции > Время окончания неактивного интервала)
and (Время обработки выбранной операции > Интервал) Then
Следующий интервал
Else if (Время окончания выбранной операции > Время окончания неактивного
интервала) and (Время обработки выбранной операции ≤ Интервал) Then
Вставка операции
End if
End For
End
```

6. Оценка эффективности

Результаты, описанные в текущем разделе, были получены на персональном компьютере с процессором Intel Core i7 3.4 ГГц и 8 ГБ оперативной памяти.

6.1. Настройка параметров

Выбор оптимальных значений параметров алгоритма оказывает большое влияние на эффективность и конечный результат. Были протестированы значения параметра α на экземпляре FT06, предложенном в [9], которые приведены в таблице 3. Значения параметров по умолчанию, определенные экспериментальным путем (более 10 запусков для каждого значения): $\beta = 0.2$, $\rho = 0.7$ для классического муравьиного алгоритма и $\beta = 1$, $\rho = 0.7$ для алгоритма системы муравьиных колоний и модифицированного муравьиного алгоритма. Наблюдение показало, что наилучшие результаты алгоритмов достигаются с помощью наборов параметров, приведенных в таблице 4.

Таблица 3. Настройка параметра α.

α	Классический муравьиный алгоритм	Алгоритм системы муравьиных колоний	Модифицированный муравьиный алгоритм
0.01	62	65	72
0.10	51	54	68
0.20	51	53	74
0.30	51	53	49
0.40	51	53	70
0.50	51	53	53
0.60	51	53	53
0.70	51	58	55
0.80	49	53	70
0.90	51	53	53
1.00	51	48	48

Таблица 4. Оптимальные наборы параметров.

Алгоритм	α	β	ρ	ε	Q_0
Классический муравьиный алгоритм	0.8	0.2	0.7	–	–
Алгоритм системы муравьиных колоний	1.0	1.0	0.7	0.7	0.8
Модифицированный муравьиный алгоритм	1.0	1.0	0.7	0.7	0.8

6.2. Оценка алгоритмов

Все три алгоритма были реализованы и запущены заданное количество раз с использованием набора наилучших параметров, которые были определены ранее. Были использованы примеры, предложенные в [10], с f от 2 до 7 включительно, то есть в общей сумме вышло 240 тестовых случаев. В качестве показателя эффективности было использовано относительное отклонение в процентах, которое рассчитывается по следующей формуле:

$$RPD = \frac{Alg - Min}{Min} \times 100, \quad (7)$$

где Alg – время выполнения, полученное текущим алгоритмом для тестового случая;

Min – наименьшее время выполнения для того же тестового случая.

Каждое значение относительного отклонения, приведенное в таблице 5, является усредненным значением при f = 2, f = 3, f = 4, f = 5, f = 6 и f = 7. Как можно заметить, модифицированный алгоритм превосходит остальные во всех без исключения случаях. Среднее значение относительного отклонения у модифицированного алгоритма равняется 0.20, в то время как у алгоритма системы муравьиных колоний данный показатель равняется 15.50, а у классического муравьиного алгоритма – 42.30 (таблица 6). Данная статистика говорит нам о том, что новый предложенный алгоритм эффективнее и надежнее своих конкурентов.

Таблица 5. Относительное отклонение протестированных алгоритмов.

№	$n \times m$	Модифицированный муравьиный алгоритм	Алгоритм системы муравьиных колоний	Классический муравьиный алгоритм
1	15 × 15	0.00	16.24	25.73
2	15 × 15	0.96	16.61	47.87
3	15 × 15	0.00	8.62	24.76
4	15 × 15	0.00	8.67	24.51
5	15 × 15	0.00	14.41	38.79
6	15 × 15	0.00	14.06	24.67
7	15 × 15	0.46	7.95	21.17
8	15 × 15	0.36	4.86	5.85
9	15 × 15	0.00	11.33	30.22
10	15 × 15	0.75	13.50	16.54
11	20 × 15	0.87	13.65	28.19
12	20 × 15	2.82	11.80	38.34
13	20 × 15	0.81	9.84	36.02
14	20 × 15	0.00	20.26	55.19
15	20 × 15	0.00	12.17	40.60
16	20 × 15	0.16	16.90	37.22
17	20 × 15	0.20	15.50	37.15
18	20 × 15	0.00	18.45	39.51
19	20 × 15	0.00	17.50	45.16
20	20 × 15	0.00	7.09	22.33
21	20 × 20	0.00	17.72	54.37
22	20 × 20	0.00	19.83	60.79
23	20 × 20	0.00	21.41	58.23
24	20 × 20	0.00	17.91	51.67
25	20 × 20	0.00	14.73	49.33
26	20 × 20	0.00	21.17	47.79
27	20 × 20	0.00	19.77	50.43
28	20 × 20	0.00	17.94	44.20
29	20 × 20	0.00	14.53	47.55
30	20 × 20	0.00	15.83	42.37
31	30 × 15	0.00	17.70	51.90
32	30 × 15	0.00	24.30	52.00
33	30 × 15	0.00	21.70	67.10
34	30 × 15	0.00	13.30	50.60
35	30 × 15	0.50	9.40	41.90
36	30 × 15	0.30	17.50	56.30
37	30 × 15	0.00	24.40	61.40
38	30 × 15	0.00	21.50	57.40
39	30 × 15	0.00	17.20	55.80
40	30 × 15	0.00	14.40	50.40

Таблица 6. Среднее относительное отклонение протестированных алгоритмов.

Алгоритм	Среднее значение
Модифицированный муравьиный алгоритм	0.20
Алгоритм системы муравьиных колоний	15.50
Классический муравьиный алгоритм	42.30

На рисунке 3 показано среднее значение относительного отклонения протестированных алгоритмов в зависимости от количества заданий. По данному графику видно, что с увеличением числа заданий, растет и разрыв между алгоритмами.

Модифицированная версия алгоритма сохраняет высокую производительность вне зависимости от количества заданий. Стоит также отметить, что классический муравьиный алгоритм и алгоритм системы муравьиных колоний справляются со своей задачей хуже при увеличении числа заданий, в то время как модифицированный алгоритм, наоборот, работает лучше.

На рисунке 4 показано среднее значение относительного отклонения протестированных алгоритмов в зависимости от количества цехов. Как и в случае с заданиями, модифицированная версия алгоритма превосходит остальные независимо от количества фабрик.

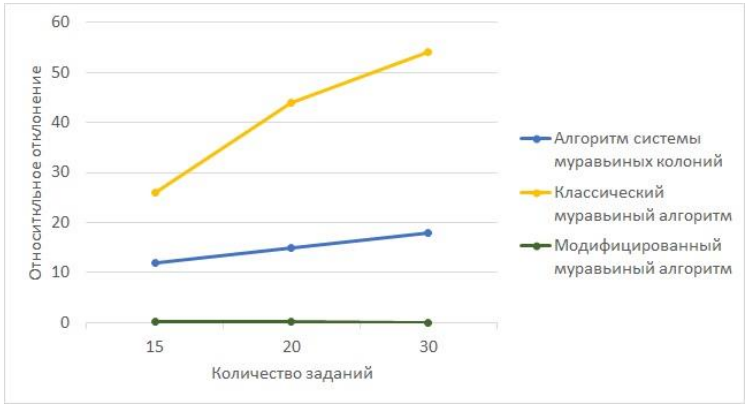

Рисунок 3. Среднее относительное отклонение в зависимости от количества заданий.

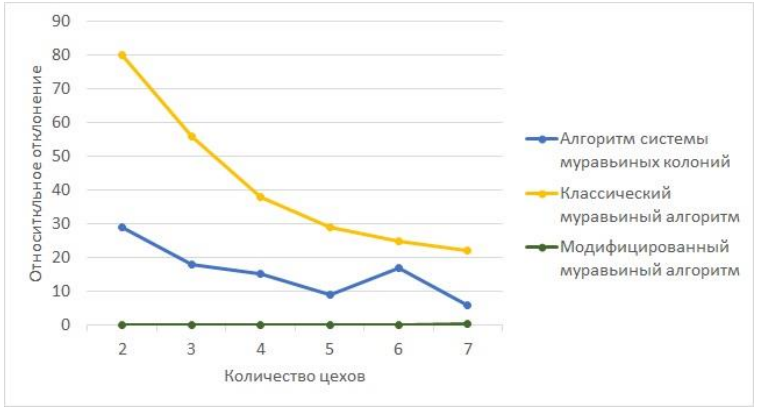

Рисунок 4. Среднее относительное отклонение в зависимости от количества цехов.

7. Заключение

Для решения задачи календарного планирования распределенных предприятий в данной статье были рассмотрены 3 версии биоинспирированного алгоритма муравьиной колонии: классический муравьиный алгоритм, алгоритм системы муравьиных колоний и модифицированный муравьиный алгоритм. Для описания модели представления календарного планирования мы использовали дизъюнктивный граф. Затем, используя правило назначения, распределили задания по соответствующим цехам. Далее, мы применили рассматриваемые алгоритмы к известным тестовым случаям из литературы. Результаты

тестирования доказали, что предложенная модификация алгоритма системы муравьиных колоний справляется с поставленной задачей гораздо эффективнее, нежели собратья.

Список литературы

[1] Jia, H. Web-based multi-functional scheduling system for a distributed manufacturing environment / H. Jia, J.Y. Fuh, A.Y. Nee, Y. Zhang // Concurrent Engineering. – 2002. – 10(1). – С. 27-39.

[2] Jia, H. A modified genetic algorithm for distributed scheduling problems / H. Jia, A.Y. Nee, J.Y. Fuh, Y. Zhang // Journal of Intelligent Manufacturin. – 2003. – 14(3-4). – С. 351-362.

[3] Jia, H. Integration of genetic algorithm and gantt chart for job shop scheduling in distributed manufacturing systems / H. Jia, J.Y. Fuh, A.Y. Nee, Y. Zhang // Computers & Industrial Engineering. – 2007. – 53(2). – С. 313-320.

[4] Chaouch, I. A survey of optimization techniques for distributed job shop scheduling problems in multi-factories / I. Chaouch, O. Belkahla Driss, K. Ghedira // In: Computer Science On-line Conference. Springer. – 2017. – С. 369-378.

[5] Błazewicz, J. The job shop scheduling problem: Conventional and new solution techniques / J. Błazewicz, W. Domschke, E. Pesch // European journal of operational research. – 1996. – 93(1). – С. 1-33.

[6] Naderi, B. Modeling and heuristics for scheduling of distributed job shops / B. Naderi, A. Azab // Expert Systems with Applications. – 2014. – 41(17). – С. 7754-7763.

[7] Dorigo, M. Optimization, learning and natural algorithms: Ph D Thesis, Politecnico di Milano/ Dorigo, M. – Italy, 1992.

[8] Talbi, E.G. Metaheuristics: from design to implementation. – vol 74 / E.G. Talbi. – John Wiley & Sons, 2009.

[9] Fisher, H. Probabilistic learning combinations of local job-shop scheduling rules / Fisher, H., G.L. Thompson // Industrial scheduling. – 1963. – 3(2). – С. 225-251.

[10] Taillard, E. Benchmarks for basic scheduling problems / E. Taillard // European Journal of Operational Research. – 1993. – 64(2). – С. 278-285.