Интеллектуальные системы управления, анализ данных

© 2019 г. О. ГОЛАМИ, канд. физ.-мат. наук (gholami-iran@yahoo.com)
(Технологический институт Блекинге, Карлскрона, Швеция),
Ю.Н. СОТСКОВ, д-р физ.-мат. наук (sotskov@newman.bas-net.by)
(Объединенный институт проблем информатики НАН Беларуси, Минск),
Ф. ВЕРНЕР, д-р наук (frank.werner@mathematik.uni-magdeburg.de)
(Университет Отто фон Герике, Магдебург, Германия),
О.С. ЗАТЮПО (ztp-oksana100@ya.ru)
(СЗАО "Серволюкс", Могилев, Беларусь)

ЭВРИСТИЧЕСКИЕ АЛГОРИТМЫ ДЛЯ МАКСИМИЗАЦИИ ДОХОДА И КОЛИЧЕСТВА ТРЕБОВАНИЙ, ОБСЛУЖИВАЕМЫХ НА ПАРАЛЛЕЛЬНЫХ ПРИБОРАХ

Множество требований необходимо обслужить на параллельных приборах. Для каждого требования известно время готовности к обслуживанию и установлен срок, не позднее которого требование должно быть обслужено. Если обслуживание требования завершается к установленному сроку, то начисляется определенная прибыль. В противном случае требование считается не обслуженным в срок, и прибыль за это требование не начисляется. Рассматривается критерий максимизации взвешенной суммы начисленной прибыли и количества требований, обслуженных в срок. Исследованы свойства целевой функции, которые позволяют строить оптимальные расписания обслуживания требований. Разработаны три эвристических алгоритма: алгоритм имитации отжига, поиск с запретами и генетический алгоритм. Разработанные программы протестированы на задачах средней размерности (50 требований и 5 приборов) и на задачах большой размерности (500 требований и 50 приборов). Даны рекомендации по использованию разработанных алгоритмов и полученных результатов в календарном планировании производства.

Ключевые слова: оптимальное расписание, параллельные приборы, максимизация прибыли, генетический алгоритм, алгоритм имитации отжига, поиск с запретами.

DOI: 10.1134/S0005231019020089

1. Введение

Планирование производства включает оптимизацию использования ограниченных ресурсов (приборов) для обслуживания заданного множества требований. Оптимальное расписание является важным фактором эффективности производства, поскольку оптимальное расписание обеспечивает сокращение расходов предприятия, уменьшение времени отклика на заявки заказчиков на продукцию предприятия, своевременное снабжение производственного

процесса сырьем и комплектующими изделиями, необходимыми для изготовления конечной продукции. Использование оптимального производственного графика позволяет уменьшить расходы на хранение сырья, комплектующих изделий и материалов и в итоге повысить эффективность использования имеющихся ресурсов и капитала, а также сократить простои оборудования. Для небольших по размерности задач оперативно-календарного планирования оптимальное расписание может быть построено с помощью универсальных точных методов оптимизации, таких как метод ветвей и границ, метод динамического программирования, методы математического программирования. Однако точные методы, как правило, не применимы для задач, которые возникают при планировании процессов для реальных промышленных предприятий, поскольку практические задачи обычно имеют большую размерность. Преобладающее множество задач оптимального планирования являются NP-трудными даже для двух или трех приборов [1]. Поэтому для решения задач оптимального планирования, возникающих на практике, обычно используют эвристические алгоритмы.

В теории расписаний рассматривается задача составления оптимального расписания, в которой каждое требование из заданного множества необходимо обслужить на одном из заданных параллельных приборов. Параллельные приборы могут быть либо идентичными, либо однотипными (однотипные приборы могут отличаться скоростями обслуживания требований). В [2] первая задача обозначается как $I//\Phi$, где Φ — целевая функция, а вторая, более сложная задача — как $Q//\Phi$. Задача $I//\Phi$ является NP-трудной даже для двух приборов при критерии минимизации общего времени работы обслуживающей системы $\Phi = \max\{C_1, \ldots, C_n\}$, где C_i — время завершения обслуживания требования J_i [2].

В данной статье рассматривается задача максимизации как дохода от обслуживания требований, так и количества требований, обслуженных в срок на параллельных неидентичных приборах. В разделе 2 приведен обзор научных работ, в которых исследовано планирование операций на параллельных приборах. В разделе 3 представлено описание исследуемой задачи и приведены результаты исследования целевой функции с экстремальными значениями. Три эвристических алгоритма описаны в разделе 4. Результаты вычислительных экспериментов с анализом разработанных алгоритмов приведены в разделе 5. Возможности применения полученных результатов на практике описаны в разделе 6. Заключительные замечания содержатся в разделе 7.

2. Обзор литературы

В [3] разработан метод нечеткого математического программирования для минимизации общего времени работы обслуживающей системы при выполнении множества операций на параллельных идентичных приборах при условии, что точные числовые данные задачи не известны при составлении расписания. Та же задача с детерминированными (точными) числовыми данными была исследована в [4]. В [5] для детерминированной обслуживающей системы разработан алгоритм “колонии муравьев” для планирования операций на параллельных приборах. Статья [6] содержит результаты выбора правил

диспетчеризации при различных критериях оптимальности (при максимизации быстродействия, минимизации суммарной продолжительности обслуживания требований и минимизации суммарных простоев приборов).

Задача планирования операций становится более сложной при рассмотрении неидентичных параллельных приборов. В [7] для такой задачи был предложен оператор обмена в целях адаптации генетического алгоритма при планировании операций на неидентичных параллельных приборах. Результаты вычислений сравнивались с результатами, полученными с помощью правила диспетчеризации по наибольшей длительности обслуживания требования (LPT). Для сведения к минимуму взвешенной суммы наименьшей длительности и наибольшей длительности обслуживания требований на неидентичных параллельных приборах в [4] был предложен метаэвристический алгоритм. В [8] представлены результаты одновременного выбора операции и назначения ее на параллельные однотипные приборы. Для такой задачи разработан алгоритм имитации отжига. Результаты приближенных вычислений сравнивались с результатами, полученными с помощью метода ветвей и границ.

В [9] генетический алгоритм был использован в имитационной модели, предложенной для максимизации быстродействия, и его результаты сравнивались с результатами, полученными при использовании правила диспетчеризации LPT. В [10] разработаны модели и предложены упрощения для неидентичных параллельных приборов при минимизации суммарной взвешенной продолжительности выполнения операций. Задача максимизации быстродействия на неидентичных параллельных приборах решалась методом целочисленного линейного программирования в [11]. Авторы статьи предложили генетический алгоритм, основанный на схеме кодирования с использованием случайных ключей. Статья [12] посвящена использованию неидентичных параллельных приборов для минимизации суммарной взвешенной продолжительности выполнения операций. В [13] исследована задача минимизации суммы раннего времени выполнения операций относительно заданных директивных сроков и запаздывания выполнения операций. Все операции были заранее упорядочены, чтобы уменьшить размерность пространства допустимых решений. Авторы предложили эвристический алгоритм для решения такой задачи. Задача с параллельными неидентичными приборами исследована в [14], где были разработаны шесть алгоритмов, основанных на поиске с запретами. Было исследовано влияние различных исходных решений на свойства полученного расписания. Эта же задача была решена с использованием алгоритма имитации отжига в [15], где было показано, что алгоритм имитации отжига более эффективен для обеспечения наилучшего использования приборов. Для задачи планирования с параллельными приборами с переналадками, зависящими от последовательности выполнения операций, и заданными временами готовности требований к обслуживанию, в [16] предложен генетический алгоритм, использованный для минимизации суммы раннего времени выполнения операций относительно заданных директивных сроков и запаздывания выполнения операций. В [17] предложен алгоритм поиска с запретами для сведения к минимуму суммарной длительности запаздываний при выполнении операций. Авторы статьи показали, что алгоритм поиска с запретами эффективнее генетического алгоритма, предложенного

в [16]. Многоцелевая модель планирования работы параллельных приборов для минимизации количества операций с запаздыванием их выполнения и суммарной продолжительности выполнения всех операций была исследована в [18] (параллельные приборы имели различные скорости выполнения операций). Задавались длительности выполнения операций и сроки их завершения, а также приоритетность выполнения заданных операций. В использованной модели была предусмотрена очередность выполнения операций в зависимости от начала их обработки и различных характеристик обслуживающих приборов. Эта задача была решена с помощью булева линейного программирования с двумя уровнями решения задачи и с использованием целочисленного линейного программирования. В [19] рассматривалась задача планирования операций, разделенных на несколько семейств для обработки их на параллельных однотипных приборах. Требовалось учитывать время настройки приборов при переходе от выполнения операций одного семейства к выполнению операций другого семейства. Было предложено несколько эвристических алгоритмов, оценена их эффективность путем сопоставления полученных результатов с нижней границей оптимального значения целевой функции и решениями, полученными с помощью точного алгоритма.

Другая важная область исследований связана с планированием работы параллельных приборов с целью максимизации дохода. В этом случае каждая операция имеет установленный срок завершения выполнения, а цель планирования состоит в том, чтобы завершить операции до установленного срока и получить от этого максимальной доход. Превышение установленного срока выполнения операции влечет за собой штраф, величина которого зависит от длительности задержки в завершении операции. В [20] предложен алгоритм имитации отжига для максимизации дохода на параллельных приборах. Эффективность алгоритма сравнивалась в вычислительных экспериментах с результатами, полученными с помощью алгоритма построения расписания согласно определенному списку операций (list scheduling) и метода ветвей и границ. Статья [21] посвящена задаче планирования выполнения параллельных операций на суперкомпьютерах с целью максимизации дохода. Авторы статьи предложили количественно оцениваемый алгоритм планирования. Было показано, что повышение производительности приборов влияет на рост доходов в большей степени, чем увеличение количества приборов.

В [22] рассматривалась задача выбора и оценки фильмов для мультиплекса с целью максимизации суммарного дохода за фиксированный период времени (горизонт планирования). Были известны времена выпуска фильмов, которые можно выбирать для проката. Если фильм выбран для его оценки, то становится известным его продолжительность, которая, вообще говоря, может увеличиваться. Авторы [22] исследовали два основных принципа составления расписаний: допустимость прерываний с последующим возобновлением обслуживания требования (preempt-resume) и запрещение таких прерываний (non-preempt). Было показано, что задача оптимизации с принципом preempt-resume является NP-трудной, в то время как задача оптимизации с принципом non-preempt полиномиально разрешима. Для решения обеих задач было разработано несколько алгоритмов. Было установлено, что доход, полученный в результате применения принципа preempt-resume, может быть

значительно больше дохода при применении принципа non-preempt. В [23] рассматривалась задача максимизации доходов провайдера. Для решения задачи применялся метод динамического распределения ресурсов на основе соглашения об уровне обслуживания, которое играет важную роль в облачных вычислениях для подключения поставщиков услуг и клиентов. Авторы [23] формализовали задачу выделения ресурсов с использованием теории массового обслуживания и предложили оптимальные решения с учетом различных параметров, таких как механизм ценообразования, скорость обслуживания и доступные ресурсы. Эксперименты показали, что разработанные алгоритмы превосходят алгоритмы, использованные ранее для решения такой задачи.

3. Постановка задачи и обозначения

Рассматриваемая задача обозначается как $Q/r_i, d_i/w_1 \sum b_i + w_2|\mathcal{J}(S)|$, если использовать трехпозиционное обозначение $\alpha/\beta/\gamma$, введенное в [2]. В этом обозначении поле α характеризует тип системы обслуживания, поле β — исходные данные и параметры, а поле γ — целевую функцию.

3.1. Постановка задачи

Задача $Q/r_i, d_i/w_1 \sum b_i + w_2|\mathcal{J}(S)|$ формулируется следующим образом. Имеется множество m параллельных однотипных приборов $\mathcal{M} = \{M_1, \ldots, M_m\}$, на которых необходимо обслужить заданное множество $\mathcal{J} = \{J_1, \ldots, J_n\}$ требований. Каждое требование $J_i \in \mathcal{J}$ может быть обслужено на любом приборе из множества $\mathcal{M}$ без прерываний. Скорости приборов $M_k \in \mathcal{M}$ могут быть различными.

Для каждого требования $J_i \in \mathcal{J}$ задано время готовности к выполнению $r_i \geqslant 0$, а также время $d_i > r_i$, к которому его выполнение должно быть завершено. Если требование $J_i \in \mathcal{J}$ выполняется целиком в период времени, принадлежащий заданному отрезку $[r_i, d_i]$, то получается прибыль $b_i > 0$. В противном случае, прибыль b_i не получается и требование J_i из расписания удаляется. Пусть $w_1 \geqslant 0$ обозначает вес полученного дохода $\sum_{J_i \in \mathcal{J}(S)} b_i$ (суммарный доход за выполнение требований при расписании S), а $w_2 \geqslant 0$ — вес количества $|\mathcal{J}(S)|$ требований $J_i \in \mathcal{J}(S) \subseteq \mathcal{J}$, выполненных при расписании S в периоды времени, принадлежащие соответствующим отрезкам $[r_i, d_i]$. Будем предполагать, что $w_1 + w_2 = 1$.

В задаче $Q/r_i, d_i/w_1 \sum b_i + w_2|\mathcal{J}(S)|$ требуется найти такое расписание S выполнения требований из множества $\mathcal{J}$ на приборах из множества $\mathcal{M}$, для которого будет максимальной сумма $w_1 \sum_{J_i \in \mathcal{J}(S)} b_i + w_2|\mathcal{J}(S)|$ взвешенной прибыли от выполнения множества $\mathcal{J}(S)$ требований и взвешенного количества $|\mathcal{J}(S)|$ требований, выполненных при расписании S в интервалах времени $[r_i, d_i]$, $J_i \in \mathcal{J}$.

Приборы из множества $\mathcal{M}$ однотипны и параллельны, иными словами, приборы могут иметь различные скорости при выполнении одного и того же требования $J_i \in \mathcal{J}$. Пусть прибор $M_k \in \mathcal{M}$ имеет скорость $v_k > 0$, тогда время p_{ik}, необходимое для выполнения требования J_i на приборе M_k, определяется следующим равенством: $p_{ik} = \frac{p_i}{v_k}$, где $p_i > 0$ обозначает *нормальное* время, необходимое для выполнения требования $J_i \in \mathcal{J}$. Пусть требование J_i

выполняется на приборе $M_k \in \mathcal{M}$ при расписании S. Выполнение требования J_i на приборе M_k может быть начато после времени r_i готовности требования к обслуживанию, а также после времени завершения выполнения $c_{lk}(S)$ предыдущего требования J_l (если оно есть в расписании S), которое выполнялось на том же приборе M_k при расписании S. Таким образом, время начала $s_{ik}(S)$ выполнения требования J_i на приборе M_k при расписании S может быть рассчитано следующим образом: $s_{ik}(S) = \max\{r_i, c_{lk}(S)\}$.

Поскольку прерывания при выполнении требований не допускаются, время $c_{ik}(S)$ завершения обслуживания требования J_i на приборе M_k при расписании S может быть вычислено как сумма времени начала s_{ik} и длительности выполнения p_{ik} требования J_i на приборе M_k, т.е. $c_{ik}(S) = s_{ik}(S) + p_{ik}$. Включение $J_i \in \mathcal{J}(S)$ имеет место, если $r_i \leqslant s_{ik}(S) < s_{ik}(S) + p_{ik} = c_{ik}(S) \leqslant d_i$. Первое слагаемое целевой функции $\Phi(S) := w_1 \sum_{J_i \in \mathcal{J}(S)} b_i + w_2|\mathcal{J}(S)|$ — это взвешенная сумма $\sum b_i(S)$ прибыли, полученной при выполнении требований из множества $\mathcal{J}$ в установленный срок: $\sum b_i(S) = \sum_{J_i \in \mathcal{J}(S)} b_i$. Второе слагаемое целевой функции $\Phi(S)$ связано с удовлетворенностью клиентов в результате увеличения количества $|\mathcal{J}(S)|$ требований $\mathcal{J}(S)$, начатых и завершенных в установленный срок при расписании S.

Если $w_1 = 1$, то $w_2 = 0$, и задача $Q/r_i, d_i/w_1 \sum b_i + w_2|\mathcal{J}(S)|$ превращается в известную задачу $Q/r_i, d_i/\sum b_i$ максимизации прибыли [20, 21].

3.2. Экстремальные значения целевой функции $\Phi(S) = w_1 \sum b_i + w_2|\mathcal{J}(S)|$

Максимально возможное значение целевой функции $\Phi(S)$ достигается в случае, когда для задачи $Q/r_i, d_i/w_1 \sum b_i + w_2|\mathcal{J}(S)|$ существует расписание S, для которого оба слагаемых $w_1 \sum_{J_i \in \mathcal{J}(S)} b_i$ и $w_2|\mathcal{J}(S)|$ целевой функции $\Phi(S)$ достигают своих максимально возможных значений, а именно: $w_1 \sum_{J_i \in \mathcal{J}(S)} b_i = w_1 \sum_{J_i \in \mathcal{J}} b_i$ и $w_2|\mathcal{J}(S)| = w_2 n$.

Теорема 1. Для задачи $Q/r_i, d_i/w_1 \sum b_i + w_2|\mathcal{J}(S)|$ существует расписание S, для которого целевая функция $\Phi(S)$ достигает своего максимально возможного значения тогда и только тогда, когда выполняется равенство $\mathcal{J}(S) = \mathcal{J}$.

Доказательство. Достаточность. Пусть для задачи $Q/r_i, d_i/w_1 \sum b_i + w_2|\mathcal{J}(S)|$ существует расписание S, для которого выполняется равенство $\mathcal{J}(S) = \mathcal{J}$. Тогда получаем следующее равенство:

$$w_1 \sum_{J_i \in \mathcal{J}(S)} b_i + w_2|\mathcal{J}(S)| = w_1 \sum_{J_i \in \mathcal{J}} b_i + w_2 n. \tag{1}$$

Очевидно, что значение $w_1 \sum_{J_i \in \mathcal{J}} b_i + w_2 n$ в правой части равенства (1) равно максимально возможному значению целевой функции $\Phi(S)$. Таким образом, расписание S, указанное в равенстве (1), обеспечивает максимально возможное значение целевой функции $\Phi(S)$. Отметим, что в этом случае расписание S является оптимальным для задачи $Q/r_i, d_i/w_1 \sum b_i + w_2|\mathcal{J}(S)|$.

Необходимость. Пусть для задачи $Q/r_i, d_i/w_1 \sum b_i + w_2|\mathcal{J}(S)|$ существует расписание S, для которого целевая функция $\Phi(S)$ достигает своего максимально возможного значения, т.е. выполняется равенство (1).

От противного предположим, что $\mathcal{J}(S) \neq \mathcal{J}$. Тогда существует требование $J_k \in \mathcal{J}$, которое не может быть обслужено в течение интервала времени $[r_k, d_k]$ при расписании S, т.е. множество $\mathcal{J}(S)$ не содержит требования J_k: $J_k \notin \mathcal{J}(S)$. Поскольку $w_1 \geqslant 0$, $w_2 \geqslant 0$ и выполняется равенство $w_1 + w_2 = 1$, то по крайней мере один вес w_1 или w_2 не равен нулю в целевой функции $\Phi(S) = w_1 \sum_{J_i \in \mathcal{J}(S)} b_i + w_2 |\mathcal{J}(S)|$.

Если $w_1 > 0$, то, используя неравенство $b_k > 0$, получаем соотношение

$$w_1 \sum_{J_i \in \mathcal{J}(S)} b_i \leqslant w_1 \sum_{J_i \in \mathcal{J}} b_i - w_1 b_k < w_1 \sum_{J_i \in \mathcal{J}} b_i.$$

Если же $w_2 > 0$, то получаем неравенства $w_2 |\mathcal{J}(S)| \leqslant w_2 (|\mathcal{J}| - 1) < w_2 n$. Таким образом, в обоих возможных случаях получаем противоречие, состоящее в том, что равенство (1) не выполняется для расписания S. Следовательно, предположение $\mathcal{J}(S) \neq \mathcal{J}$ не верно, что и требовалось доказать.

В приведенном доказательстве теоремы 1 получено следующее утверждение.

Следствие 1. Если выполняется равенство $\mathcal{J}(S) = \mathcal{J}$, то расписание S является оптимальным для задачи $Q/r_i, d_i/w_1 \sum b_i + w_2 |\mathcal{J}(S)|$.

Следующая теорема характеризует другое экстремальное значение целевой функции $\Phi(S)$, а именно, минимально возможное значение функции $\Phi(S)$.

Теорема 2. Оптимальное значение целевой функции $\Phi(S)$ равно нулю для задачи $Q/r_i, d_i/w_1 \sum b_i + w_2 |\mathcal{J}(S)|$ тогда и только тогда, когда не существует требования $J_i \in \mathcal{J}$, которое может быть полностью обслужено в течение заданного интервала времени $[r_i, d_i]$.

Доказательство. Достаточность. Предположим, что не существует требования $J_i \in \mathcal{J}$, которое может быть обслужено в течение заданного интервала времени $[r_i, d_i]$. Из этого следует, что должно выполняться равенство $\mathcal{J}(S) = \emptyset$ для любого допустимого расписания S. Следовательно, оптимальное значение целевой функции $\Phi(S)$ равно нулю (т.е. минимально возможному значению целевой функции $\Phi(S)$):

$$\Phi(S) = w_1 \sum_{J_i \in \mathcal{J}(S)} b_i + w_2 |\mathcal{J}(S)| = w_1 0 + w_2 0 = 0.$$

Необходимость. Пусть оптимальное значение целевой функции $\Phi(S)$ равно нулю для рассматриваемой задачи $Q/r_i, d_i/w_1 \sum b_i + w_2 |\mathcal{J}(S)|$. От противного предположим, что существует требование $J_i \in \mathcal{J}$, которое может быть полностью обслужено в течение интервала времени $[r_i, d_i]$. Следовательно, существует расписание S, для которого требование $J_i \in \mathcal{J}$ обслуживается в течение интервала времени $[r_i, d_i]$. Далее оценим значение целевой функции $\Phi(S)$ для такого расписания S. Поскольку $w_1 \geqslant 0$, $w_2 \geqslant 0$ и равенство $w_1 + w_2 = 1$ выполняется, то по крайней мере один вес w_1 или w_2 не равен нулю для целевой функции $\Phi(S) = w_1 \sum_{J_i \in \mathcal{J}(S)} b_i + w_2 |\mathcal{J}(S)|$.

Если $w_1 > 0$, то, используя неравенство $b_i > 0$, получаем $\Phi(S) \geqslant$ $\geqslant w_1 \sum_{J_i \in \mathcal{J}(S)} b_i \geqslant w_1 b_i > 0$. Если же $w_2 > 0$, то получаем $\Phi(S) \geqslant w_2 |\mathcal{J}(S)| \geqslant$

$\geqslant w_2 > 0$. Таким образом, в обоих случаях есть противоречие тому, что оптимальная величина целевой функции $\Phi(S)$ равна нулю. Это означает, что предположение о том, что существует требование $J_i \in \mathcal{J}$, которое может быть полностью обслужено в течение интервала времени $[r_i, d_i]$, опровергнуто, что и требовалось доказать.

Покажем, что проверка условия теоремы 2 может быть реализована за время $O(n)$ при естественном предположении о том, что $n \geqslant m$. Действительно, для проверки условия теоремы 2 достаточно выбрать прибор M_z с максимальной скоростью $v_z = \max\{v_k \mid M_k \in \mathcal{M}\}$ и для требований J_i из множества $\mathcal{J}$ проверить неравенство

$$\frac{p_i}{v_z} \leqslant d_i - r_i. \tag{2}$$

Если хотя бы для одного требования из множества $\mathcal{J}$ неравенство (2) выполняется, то для задачи $Q/r_i, d_i/w_1 \sum b_i + w_2|\mathcal{J}(S)|$ существует расписание S, для которого $\Phi(S) > 0$. В противном случае равенство $\Phi(S) = 0$ выполняется для любого расписания S, существующего для задачи $Q/r_i, d_i/w_1 \sum b_i + w_2|\mathcal{J}(S)|$.

4. Эвристические алгоритмы для задачи $Q/r_i, d_i/w_1 \sum b_i + w_2|\mathcal{J}(S)|$

В этом разделе описываются алгоритмы имитации отжига, поиска с запретами и генетический алгоритм для приближенного решения задачи $Q/r_i, d_i/w_1 \sum b_i + w_2|\mathcal{J}(S)|$. Применение этих алгоритмов для решения задачи $Q/r_i, d_i/w_1 \sum b_i + w_2|\mathcal{J}(S)|$ может начинаться с любого эвристического решения задачи (допустимого расписания). Исследование различных аспектов разработанных алгоритмов (таких как число итераций, число соседей, количество поколений построенных расписаний и время реализации алгоритма) позволило получить достаточно большие значения целевой функции $\Phi(S) = w_1 \sum b_i + w_2|\mathcal{J}(S)|$ для задач средней размерности.

4.1. Генетический алгоритм

Генетический алгоритм является эволюционным алгоритмом, представляющим решение (индивидуум) задачи в виде кода, называемого *хромосомой* или *геномом*. Аналогично процессу эволюции в живой природе хромосомы *скрещиваются* и *мутируют* для создания новых индивидуумов в каждом поколении. Новые хромосомы создаются путем присоединения сегментов, выбранных поочередно от каждой из хромосом двух родителей, которые имеют фиксированную длину. При таком процессе, соединив лучшие хромосомы лучших индивидуумов, можно получить лучший индивидуум. Оператором обмена выбирается несколько частей текущего поколения, и в геноме будут происходить определенные трансформации.

При решении задачи $Q/r_i, d_i/w_1 \sum b_i + w_2|\mathcal{J}(S)|$ n требований множества $\mathcal{J}$ должны быть назначены на приборы из множества $\mathcal{M}$. Последовательность геномов g_k в хромосоме $X_i = [g_1 g_2 g_3 \dots g_u]$ определяет упорядоченное

множество требований, которые обслуживаются на приборе $M_i \in \mathcal{M}$ в указанном порядке. Различные времена готовности требований к обслуживанию и заданные директивные сроки завершения их обслуживания могут привести к возникновению конфликтов при обслуживании требований из хромосомы X_i. Например, рассмотрим хромосому $X_i = [g_6 g_7 g_3 g_{10} g_5 g_9 g_8 g_{11} g_1 g_2 g_4]$ в качестве упорядоченного списка требований, которые должны быть обслужены на приборе M_i. Требования должны обслуживаться на приборе M_i в порядке слева направо при условии, что не нарушаются заданные времена готовности требований к обслуживанию и директивные сроки их обслуживания. Пусть $Y_i = \{g_6 g_7 g_{10} g_9 g_8 g_{11} g_2 g_4\}$ обозначает множество обслуженных в срок требований, а $Z_i = \{g_3 g_5 g_1\}$ — множество требований, которые не были обслужены в срок, $X_i = Y_i \cup Z_i$. Для того, чтобы обеспечить возможность одновременного обслуживания требований на m параллельных приборах из множества $\mathcal{M}$, используются разделители d. Разделители определяют разбиение последовательности X_i на подпоследовательности требований, назначенных на различные приборы. Для того чтобы назначить требования на m параллельных приборов, необходимо использовать $m-1$ разделителей $d = \{d_1 d_2 \dots d_{m-1}\}$. Разделители добавляются в хромосому X_i случайно по равномерному закону распределения вероятностей. Рассматривая хромосому $X_i = [g_6 g_7 g_3 g_{10} g_5 g_9 g_8 g_{11} g_1 g_2 g_4]$ для назначения требований на два (три) прибора, необходим один разделитель (два разделителя). Для случая двух приборов M_1 и M_2 можно получить случайную последовательность $X_a = [g_6 g_7 g_3 g_{10} d_1 g_5 g_9 g_8 g_{11} g_1 g_2 g_4]$, определяющую две подпоследовательности $\{g_6 g_7 g_3 g_{10}\}$ и $\{g_5 g_9 g_8 g_{11} g_1 g_2 g_4\}$ параллельного обслуживания требований на двух приборах M_1 и M_2 соответственно. Для случая трех приборов M_1, M_2 и M_3 можно получить случайную последовательность $X_b = [g_6 g_7 d_1 g_3 g_{10} g_5 g_9 d_2 g_8 g_{11} g_1 g_2 g_4]$, определяющую три подпоследовательности $\{g_6 g_7\}$, $\{g_3 g_{10} g_5 g_9\}$ и $\{g_8 g_{11} g_1 g_2 g_4\}$ параллельного обслуживания требований на трех приборах M_1, M_2 и M_3 соответственно.

Рассмотрим две хромосомы $X_a = [g_6 g_7 g_3 g_{10} d_1 g_5 g_9 g_8 g_{11} g_1 g_2 g_4]$ и $X_c = [g_8 g_7 g_1 g_9 g_3 d_1 g_5 g_{10} g_6 g_2 g_4]$, определяющие случайно сгенерированные последовательности обслуживания требований на двух приборах M_1 и M_2 соответственно. Оператор скрещивания работает следующим образом. Выбирается случайно позиция в хромосоме $X_a = [g_6 g_7 g_3 g_{10} d_1 g_5 g_9 g_8 g_{11} g_1 g_2 g_4]$. Например, выбран геном g_9. Тогда первая часть $[g_6 g_7 g_3 g_{10} d_1 g_5 g_9]$ хромосомы X_a копируется в порождаемую хромосому $X_e = [g_6 g_7 g_3 g_{10} d_1 g_5 g_9]$. Затем последовательно просматриваются геномы хромосомы X_c, начиная с первого генома g_8. Если очередной геном g_k не содержится в уже сгенерированной части новой хромосомы $X_e = [g_6 g_7 g_3 g_{10} d_1 g_5 g_9 \dots]$, то геном g_k добавляется в новую хромосому в том же порядке, как и в хромосоме X_c. В рассматриваемом примере получается следующая новая хромосома $X_e = [g_6 g_7 g_3 g_{10} d_1 g_5 g_9 g_8 g_1 g_2 g_4]$. Заметим, что в построенной хромосоме X_e изменился порядок требований и разделителей. Для оператора мутации случайно выбираются два генома в рассматриваемой хромосоме (при этом разделитель также может быть выбран). Затем выбранные геномы (или геном и разделитель) меняются местами в новой хромосоме. Например, следующая хромосома $X_e = [g_6 g_7 g_3 g_4 d_1 g_5 g_9 g_8 g_1 g_2 g_{10}]$ получена из хромосомы X_e в результате перемены мест геномов g_{10} и g_4.

Алгоритм 1 представляет собой псевдокод генетического алгоритма, разработанного для эвристического решения задачи $Q/r_i, d_i/w_1 \sum b_i + w_2|\mathcal{J}(S)|$.

Алгоритм 1. Псевдокод генетического алгоритма

Require: Максимальное число ($itr_{\max}$) итераций; количество хромосом в каждом поколении (n_{pop}); число $m = N$ приборов; список требований J; X — процент скрещиваний; Y — процент мутаций; Z — процент частей для нового поколения.
$n_{itr} \leftarrow 1$ // счетчик генераций
$C_{n_1} = null$ // $C_{n_{itr}}$ — множество хромосом в генерации n_{itr}
for i=1 **to** n_{pop} **do**
$q_c \leftarrow$ случайное решение (J)
$C_{n_{itr}} \leftarrow C_{n_{itr}} \cup (q_c, evaluate(q_c))$
end for
while $(n_{itr} \leqslant itr_{\max})$ **do**
$n_{itr} \leftarrow n_{itr} + 1$
for i=1 **to** $n_{pop} \times X\%$ **do**
$q_c \leftarrow$ crossover(roulette-wheel($C_{n_{itr}-1}$), roulette-wheel $(C_{n_{itr}-1})$)
$C_{n_{itr}} \leftarrow C_{n_{itr}} \cup (q_c, evaluate(q_c))$
end for
for i=1 **to** $n_{pop} \times Y\%$ **do**
$q_c \leftarrow$ mutate(roulette-wheel($C_{n_{itr}-1}$))
$C_{n_{itr}} \leftarrow C_{n_{itr}} \cup (q_c, evaluate(q_c))$
end for
$C_{n_{itr}} \leftarrow C_{n_{itr}} \cup elites(C_{n_{itr}-1}, Z\%)$ // добавляем в новую совокупность $C_{n_{itr}}$ $Z\%$ лучших $C_{n_{itr}-1}$ хромосом как элитные
if лучшее решение не изменяется ($C_{n_{itr}}$, $itr_{\max} \times 10\%$) **then**
$exit$
end if
end while
return лучшее решение (n_{itr})

В генетическом алгоритме 1 используются следующие параметры: количество хромосом в первом поколении n_{pop}, вероятность операции скрещивания p_c, вероятность операции мутации p_m, максимальное число допустимых поколений $itr_{\max}$. В данном эксперименте количество хромосом в первом поколении составило 500, вероятность операции скрещивания $p_c = 0{,}6$, вероятность операции мутации $p_m = 0{,}04$, максимальное число итераций 1000. Первое поколение состоит из n_{pop} хромосом. Длина хромосомы n_{var} равна сумме количества требований и количества приборов минус один: $n_{var} = n + m - 1$. Числа между 1 и n_{var} были случайно распределены в хромосоме по равномерному закону распределения вероятностей.

После первого шага алгоритма вычисляется функция приспособленности. Для выбора родителей из популяции используется функция “колесо рулетки”. В разработанном генетическом алгоритме используется двухточечная мутация. После скрещивания используется функция исправления ошибок для

преобразования поврежденных хромосом. Для мутации используется функция замены, т.е. случайным образом выбираются два места в хромосоме и обмениваются коды внутри этих мест. В хромосоме натуральные числа от 1 до n представляют собой индекс i для соответствующего требования $J_i \in \mathcal{J}$, $1 \leqslant i \leqslant n$. Если число g в хромосоме больше n, $g > n$, то число g определяет индекс k используемого прибора M_k следующим образом: $k = g - n$. Наибольший ген ограничен числом $n + m$.

Функция приспособленности вычисляется следующим образом: $Fitness = w_1 \sum_{J_i \in \mathcal{J}(S)} b_i + w_2 |\mathcal{J}(S)|$, что соответствует значению целевой функции $\Phi(S)$. Генетический алгоритм выполняется до заданного максимального числа поколений ($itr_{\max}$). Однако если полученное рекордное расписание не улучшается в течение определенного числа построенных поколений, то генетический алгоритм может быть также остановлен. В вычислительных экспериментах использовался вес $w_1 = 0{,}7$ для суммарного дохода и вес $w_2 = 1 - w_1 = 0{,}3$ для количества $|\mathcal{J}(S)|$ требований, обслуженных в срок при расписании S.

4.2. Алгоритм имитации отжига

Благодаря своей простоте, алгоритм имитации отжига часто используется для решения задач оптимизации. Этот алгоритм начинается с одной точки в пространстве решений задачи, т.е. с расписания S, полученного каким-либо эвристическим алгоритмом. Затем исследуется пространство решений задачи для того, чтобы найти лучшее расписание S' [26]. Если построенное расписание S' будет иметь большее значение целевой функции, чем предыдущее рекордное (лучшее из построенных) расписание S, то расписание S' будет использоваться как новое рекордное расписание ($S \longleftarrow S'$). Если же новое расписание S' не лучше расписания S, то расписание S будет приниматься с вероятностью $P(S, S', T)$ в качестве рассматриваемого расписания. После каждого шага система отжига "остывает", уменьшая "температуру", и принятие нового расписания к рассмотрению происходит с вероятностью $P(S, S', T)$. В алгоритме имитации отжига используются следующие параметры: количество новых соседей NN за одну итерацию; начальная температура T_0; конечная температура T_f; процедура для уменьшения температуры T_{rf}; максимальное количество итераций $itr_{\max}$, разрешенных для реализации алгоритма имитации отжига. Для создания новых соседей рассматриваемой перестановки используются процедуры обмена. После создания NN соседей лучшее из полученных расписаний будет выбрано как новое расписание S', которое будет рассматриваться далее. При этом вероятность принятия нового расписания $P(S, S', T)$ определяется следующим образом: $P_r = e^{(-E)/T}$.

Алгоритм 2 представляет собой псевдокод алгоритма имитации отжига для решения задачи $Q/r_i, d_i/w_1 \sum b_i + w_2 |\mathcal{J}(S)|$.

Алгоритм 2. Псевдокод алгоритма имитации отжига

Require: Количество новых соседей NN; начальная температура T_0; конечная температура T_f; максимальное число итераций $itr_{\max}$; количество $m = N$ приборов; список требований J.

$n_{itr} \leftarrow 1$ // счетчик итераций
$T \leftarrow T_0$
$S \leftarrow$ случайно найденное решение (J)
while $(n_{itr} \leqslant itr_{\max})$ **do**
  $n_{itr} \leftarrow n_{itr} + 1$
  $S' \leftarrow \mathrm{swap}(S)$ // возврат расписания (S)
  $\Delta F \leftarrow (Fit(S') - Fit(S))/Fit(S)$
  if $(\Delta F < 0)$ **then**
    $S \leftarrow S'$
  else
    $r \leftarrow random(0, 1)$
    $P_r \leftarrow e^{(-\Delta F)/T}$
    if $(r < P_r)$ **then**
      $S \leftarrow S'$
    end if
  end if
  $T_{rf} = (T_0 - T_f)/itr_{\max}$
  $T = T - T_{rf}$ // контроль за температурой
  if not-changed$(S, itr_{\max} \times 10\%)$ **then**
    $exit$
  end if
end while
return лучшее решение S

После реализации очередного шага алгоритма случайным образом выбирается действительное число в промежутке от нуля до единицы по равномерному закону распределения вероятностей. Если это число меньше P_r, то построенное расписание будет принято. После завершения описанного шага температура будет понижена до значения $T := T - T_{rf}$, где $T_{rf} = (T_0 - T_f)/itr_{\max}$ определяет величину, на которую произойдет снижение температуры. Реализация алгоритма продолжается до максимального числа $itr_{\max}$ разрешенных итераций. Если полученное наилучшее расписание не улучшается в пределах заданного количества итераций, то алгоритм имитации отжига может быть остановлен.

4.3. Алгоритм поиска с запретами

Алгоритм поиска с запретами является мета-эвристическим алгоритмом, предложенным в [24] в 1986 г. Выполнение алгоритма начинается с эвристически построенного расписания S. Как и алгоритм имитации отжига, алгоритм поиска с запретами определяет наилучшее соседнее расписание S' относительно рассматриваемого расписания. Если новое расписание S' не входит в список расписаний с запретами, то оно будет выбрано как текущее решение: $S \leftarrow S'$. В противном случае будет проверено следующее условие на принятие расписания S'. Если новое соседнее расписание S' лучше, чем текущее расписание S, то оно будет принято как новое решение S', несмотря на то, что расписание S' может попасть в список расписаний с запретами. Фактически,

список запретов используется для выхода из локального оптимума. Если новый переход добавляется в список запретов, то некоторые другие переходы могут быть удалены из него. Параметр, называемый списком запретов TL, задает количество итераций, при выполнении которых должен быть изменен список запретов. Такой переход к новому соседнему расписанию будет продолжаться до тех пор, пока не будет выполнено условие остановки алгоритма. После реализации каждого шага алгоритма список запретов будет обновляться путем добавления нового запрещенного перехода с целью предотвращения возврата алгоритма к предыдущему решению (т.е. для исключения зацикливания алгоритма поиска с запретами).

Алгоритм 3 представляет собой псевдокод поиска с запретами для решения задачи $Q/r_i, d_i/w_1 \sum b_i + w_2|\mathcal{J}(S)|$.

Алгоритм 3. Псевдокод алгоритма поиска с запретами

Require: Количество NN соседних перестановок; максимальное число lim_t запрещенных переходов в списке запретов; максимальное число реализуемых итераций $itr_{\max}$; количество $m = N$ приборов; список требований J.
$n_{itr} \leftarrow 1$; // счетчик выполненных итераций
set neighbors $\leftarrow NULL$ // множество соседей
$S_{now} \leftarrow$ случайно найденное решение (J)
clear-history(H)
while $(n_{itr} \leqslant itr_{\max})$ **do**
  $n_{itr} \leftarrow n_{itr} + 1$
  for i=1 **to** NN **do**
    $S' \leftarrow$ swap(S_{now}) // обратная перестановка (S_{now})
    $neighbors \leftarrow neighbors \cup S'$
  end for
  for all $S_{candid} \in neighbors$ **do**
    if $(Fit(S_{candid}) < Fit(S_{new}))$ and $(S_{candid} \notin H)$ **then**
      $S_{new} \leftarrow S_{candid}$
    end if
  end for
  if $(Fit(S_{new}) < Fit(S_{now}))$ **then**
    $S_{now} \leftarrow S_{new}$
    update-history(H, S_{now})
  end if
  if not-changed$(S_{now}, itr_{\max} \times 10\%)$ **then**
    exit
  end if
end while
return лучшее решение S_{now}

Число соседних перестановок NN за одну итерацию, ограничение переходов в списке lim_t и максимальное число реализуемых итераций $itr_{\max}$ — это три основных параметра в алгоритме поиска с запретами. Запреты на переходы — это переходы, которые уже были реализованы на предшествующих ите-

рациях. Если алгоритм поиска с запретами достиг максимально возможного количества переходов в построенном списке запретов (tabu list), то переходы с индексами больше единицы, $TL > 1$, становятся запрещенными для новых переходов.

Если в список запретов на перемещения был добавлен переход, то параметр lim_t показывает, что в следующих lim_t итерациях этот переход будет запрещен. Например, назначение $lim_t = 5$ для какого-то перехода будет означать, что в следующих пяти итерациях алгоритма этот переход будет запрещен. В нашем алгоритме поиска с запретами предполагалось, что $NN = 1000$, $itr_{\max} = 1000$, а параметр lim_t вычислялся следующим образом: $lim_t = \lceil itr_{\max} \times 0.05 \rceil$.

Исходное решение в алгоритме поиска с запретами генерируется каким-либо эвристическим алгоритмом. Затем алгоритм поиска с запретами создает несколько соседей (путем применения функции замены). После создания NN соседних перестановок выбираются лучший разрешенный сосед и лучший запрещенный сосед. Если лучший разрешенный сосед лучше, чем лучший запрещенный сосед, то лучший разрешенный сосед будет выбран как новое решение (даже если оно не лучше текущего рекордного решения). С другой стороны, если лучший запрещенный сосед лучше, чем лучший разрешенный сосед и лучшее построенное рекордное решение, то лучший запрещенный сосед лучше и будет принят в качестве нового решения. Если все созданные решения находятся в списке запретов, то лучший сосед будет выбран в качестве нового решения. После замещения текущего рекордного решения его соседом список запретов должен быть обновлен. Это означает, что предыдущий выполненный переход, в результате которого получилось новое решение, будет добавлен в список запретов. Это добавление перехода в список запретов позволяет алгоритму предотвратить возвращение текущего решения в прежнее состояние. При этом индексы всех переходов в списке запретов уменьшают на единицу: $TL_i := \max\{TL_i - 1,\ 0\}$. После такого изменения новый переход будет добавлен в список запретов со следующим индексом, рассматриваемым как ограничение: $TL_k(sol_{m1}, sol_{m2}) = Limit$; $TL_l(sol_{m2}, sol_{m1}) = Limit$.

Выполнение алгоритма 3 продолжается до тех пор, пока не будет достигнуто максимальное количество допустимых итераций $itr_{\max}$. Если рекордное расписание не улучшается после заданного количества итераций, то алгоритм будет остановлен.

5. Вычислительный эксперимент

Для того чтобы оценить эффективность разработанных алгоритмов (генетического алгоритма, алгоритма имитации отжига и алгоритма поиска с запретами), было сгенерировано случайным образом 20 примеров задачи $Q/r_i, d_i/w_1 \sum b_i + w_2|\mathcal{J}(S)|$ с одинаковой средней размерностью $n \times m$, где $n = 50$ и $m = 5$. Для сравнения разработанных алгоритмов решения задач $Q/r_i, d_i/w_1 \sum b_i + w_2|\mathcal{J}(S)|$ большой размерности, которые могут возникать на практике, случайным образом были сгенерированы 10 задач с количеством требований $n = 500$ и количеством приборов $m = 50$.

Таблица 1. Характеристики входных данных четырех классов задач с 50 требованиями и 5 приборами

1	2	3	4	5
Класс задач	Длительность обслуживания требования	Время готовности требования к обслуживанию	Директивный срок обслуживания требования	Прибыль от обслуживания требования
1	[1–10]	[0–40]	[1–10]	[1–20]
2	[1–10]	[0–70]	[1–10]	[1–20]
3	[1–10]	[0–40]	[1–20]	[1–20]
4	[1–10]	[0–70]	[1–20]	[1–20]

Задачи средней размерности были пронумерованы последовательно: $1, 2, \ldots, 20$. Задачи большой размерности пронумерованы последовательно: $21, 22, \ldots, 30$. Входные данные для задач средней размерности состоят из $n = 50$ требований $J_i \in \mathcal{J} = \{J_1, \ldots, J_{50}\}$, которые должны быть обслужены на $m = 5$ параллельных однотипных приборах. Скорости приборов $v_k \in \{0{,}5; 0{,}8; 1; 1{,}1; 1{,}3\}$ выбирались случайным образом для приборов $M_k \in \mathcal{M} = \{M_1, \ldots, M_5\}$. В табл. 1 указаны интервалы генерации длительностей обслуживания требований $J_i \in \{J_1, \ldots, J_{50}\}$ (столбец 2), интервалы генерации времени r_i готовности требований $J_i \in \mathcal{J}$ к обслуживанию (столбец 3), интервалы генерации директивных сроков d_i окончания обслуживания требований $J_i \in \mathcal{J}$ (столбец 4), а также интервалы генерации прибыли, полученной от своевременного обслуживания требований $J_i \in \mathcal{J}$ (столбец 5).

Наборы входных данных для 20 случайно сгенерированных задач $1, \ldots, 20$ средней размерности можно разделить на четыре класса в соответствии с их характеристиками. В первом классе задач (примеры 1–5) времена готовности требований к обслуживанию и директивные сроки окончания обслуживания требований были сгенерированы достаточно близкими между собой. Во втором классе задач (примеры 6–10) времена готовности требований к обслуживанию были сгенерированы относительно далекими одно от другого, а директивные сроки окончания обслуживания требований были сгенерированы более близкими. В третьем классе задач (примеры 11–15) времена готовности требований к обслуживанию были сгенерированы достаточно близкими, а директивные сроки окончания обслуживания требований были сгенерированы более далекими одно от другого. В четвертом классе задач (примеры 16–20) времена готовности требований к обслуживанию и директивные сроки окончания обслуживания требований были сгенерированы достаточно далекими одно от другого. Длительности обслуживания требований случайным образом выбирались из множества $\{1, \ldots, 10\}$. Времена готовности требований к обслуживанию генерировались случайным образом в пределах разрешенных для них интервалов, которые указаны в табл. 1. Директивный срок окончания обслуживания требований определялся как сумма времени готовности требования к обслуживанию, длительности обслуживания требования и случайного числа random.range из допустимого интервала положительной длины: $d_i = r_i + p_i + random.range$. Отсюда следует, что условие теоремы 2 не выполняется ни для одного из сгенерированных примеров, решенных в вычислительных экспериментах. Поэтому оптимальное значение целевой функ-

ции $\Phi(S)$ было строго положительным, $\Phi(S) > 0$, для любого из решенных примеров задачи $Q/r_i, d_i/w_1 \sum b_i + w_2|\mathcal{J}(S)|$. Диапазоны возможных параметров для задач перечисленных классов представлены в табл. 1.

Вычислительные эксперименты были реализованы на компьютере со следующими характеристиками: Pentium 4, Dual Core, 1,8 ГГц и Рам 4 ГБ.

5.1. Определение приемлемых параметров разработанных алгоритмов

Различные свойства разработанных алгоритмов такие, как число итераций, количество соседей, размер популяции, время работы компьютера (CPU-time) и качество построенных расписаний, исследовались в предварительных экспериментах для того, чтобы увеличить значения целевой функции для расписаний, построенных с помощью разработанных алгоритмов для задач $1, \ldots, 30$. Результаты предварительных вычислительных экспериментов по применению генетического алгоритма показали, что увеличение размера популяции (n_{pop}) и вероятности мутаций (p_m) положительно влияет на качество построенных расписаний. Увеличение размера популяции n_{pop} до 1000 родителей может улучшить качество построенных расписаний. Однако после достижения определенного предела в количестве построенных поколений данный фактор уже не имел существенного эффекта. На рис. 1 представлены первое слагаемое ($Obj1 = w_1 \sum b_i$) и второе слагаемое ($Obj2 = w_2|\mathcal{J}(S)|$) целевой функции $\Phi(S) = w_1 \sum b_i + w_2|\mathcal{J}(S)| = Obj1 + Obj2$ для экспериментов с различными входными параметрами.

$(w_1,w_2) = (1,0)$, $n_{pop} = 100$, $p_c = 0{,}4$, $p_m = 0{,}6$

*Obj*1	*Obj*2
415	34
406	32
416	32
398	30
361	27
402	30
424	33
413	33
418	32
428	33

$(w_1,w_2) = (0,1)$, $n_{pop} = 1000$, $p_c = 0{,}6$, $p_m = 0{,}8$

*Obj*1	*Obj*2
430	40
443	44
429	44
423	44
395	39
471	44
392	40
420	40
439	44
425	43

$(w_1,w_2) = (0,1)$, $n_{pop} = 1000$, $p_c = 0{,}8$, $p_m = 1$

*Obj*1	*Obj*2
483	46
481	45
427	43
425	42
450	45
448	44
450	42
488	46
476	45
453	43

$(w_1,w_2) = (0{,}7;0{,}3)$, $n_{pop} = 1000$, $p_c = 1$, $p_m = 1$

*Obj*1	*Obj*2
507	47
470	39
499	44
478	40
492	42
488	45
496	43
488	46
476	45
496	43

$(w_1,w_2) = (1,0)$, $n_{pop} = 1000$, $p_c = 1$, $p_m = 1$

*Obj*1	*Obj*2
491	42
492	42
491	42
476	38
498	44
487	43
507	47
498	43
491	43
502	46

Рис. 1. Качество расписаний с различными исходными параметрами для генетического алгоритма.

Таблица 2. Результаты решений примеров с диапазонами $(w_1, w_2) = (1, 0)$ и $(w_1, w_2) = (0, 1)$, полученных с помощью алгоритма имитации отжига при $n_{itr} = 100$, $NN = 100$, $T_0 = 100$ и $T_f = 0$

$(w_1, w_2) = (1, 0)$				$(w_1, w_2) = (0, 1)$			
$\sum b_i$	J_{ok}	CPU-time	n_{itr}	$\sum b_i$	J_{ok}	CPU-time	n_{itr}
502	45	2572	539	495	47	2802	548
496	43	2485	638	456	45	1729	384
499	44	4985	661	455	46	1901	436
495	43	1543	389	472	46	1421	335
499	44	2005	500	455	45	1627	406

Для настройки алгоритма имитации отжига количества NN соседей, рассматриваемых на каждой итерации алгоритма, начинались с 5, а допустимые количества итераций n_{itr} начинались с 10. Большое число повторных применений алгоритма имитации отжига показало, что лучшие расписания получаются при количестве соседей $NN = 100$ и при допустимом количестве итераций, равном $n_{itr} = 100$. Было установлено, что при увеличении этих параметров качество расписаний существенно не улучшается, а время работы алгоритма значительно увеличивается.

На основе результатов предварительных вычислительных экспериментов было установлено, что существует значительная корреляция между двумя слагаемыми $w_1 \sum b_i$ и $w_2|\mathcal{J}(S)|$ целевой функции $\Phi(S) = w_1 \sum b_i + w_2|\mathcal{J}(S)|$ при условии, что веса w_1 и w_2 имеют близкие значения. Это означает, что при увеличении числа $|\mathcal{J}(S)| =: J_{ok}$ требований $J_i \in \mathcal{J}(S)$, обслуженных в течение заданных интервалов времени $[r_i, d_i]$ при построенном расписании S, следует ожидать и увеличение прибыли, полученной за выполнение множества требований в срок. Аналогично, при увеличении прибыли увеличивается и число J_{ok} требований во множестве $\mathcal{J}(S)$ при условии, что веса w_1 и w_2 имеют достаточно близкие значения.

Указанная корреляция существенно слабеет, если веса w_1 и w_2 значительно отличаются один от другого. В табл. 2 представлены результаты эксперимента для $n_{itr} = 100$, $NN = 100$, $(w_1, w_2) = (1, 0)$ и $(w_1, w_2) = (0, 1)$. Сравнивая полученные результаты, можно заметить, что большую прибыль можно получить, если большинство требований $J_i \in \mathcal{J}$ принадлежат множеству $\mathcal{J}(S)$ при расписании S.

5.2. Обсуждение полученных вычислительных результатов

Максимальное число $itr_{\max}$ разрешенных итераций (поколений) эвристического алгоритма является важным фактором качества наилучшего из построенных расписаний. Проведенные вычислительные эксперименты показали, что для генетического алгоритма после 200 поколений значение целевой функции $\Phi(S) = w_1 \sum b_i + w_2|\mathcal{J}(S)|$ для наилучшего построенного расписания S не увеличивается (см. рис. 2). Для двух других разработанных алгоритмов существует аналогичная связь, т.е. увеличение числа разрешенных итераций $itr_{\max}$, существенно не влияет на качество наилучших из полученных расписаний, но при этом может существенно увеличиваться время реа-

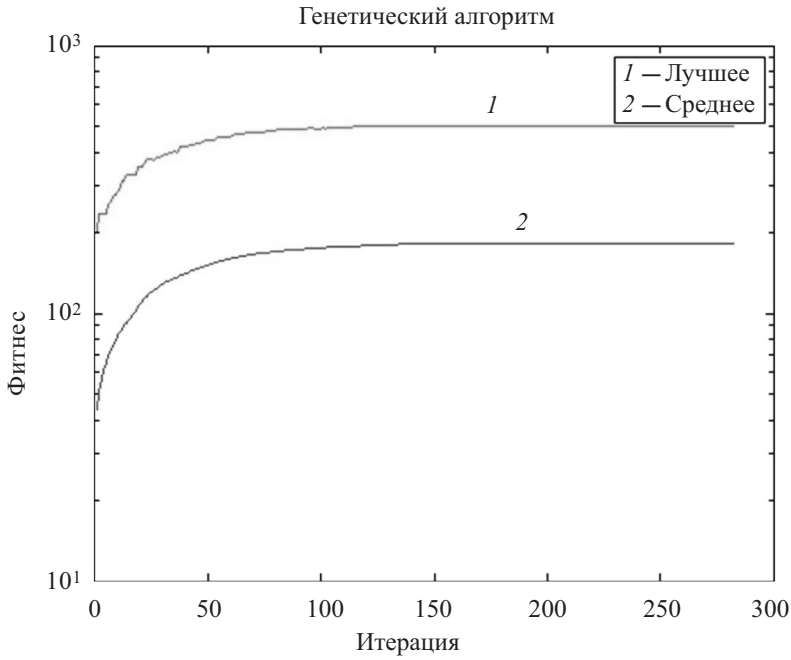

Рис. 2. Значения целевой функции после каждой итерации генетического алгоритма.

лизации алгоритма (CPU-time). В эксперименте исследовалась скорость изменения значений целевой функции $\Phi(S)$ после каждой итерации алгоритма. Большое число применений разработанных алгоритмов показало, что подходящее количество итераций для алгоритмов имитации отжига и поиска с запретами находится около 500 (см. рис. 3 и 4).

На рис. 4 для алгоритма поиска с запретами среднее и лучшее значения целевой функции одинаковы, поскольку в этом алгоритме используется только одно решение задачи $Q/r_i, d_i/w_1 \sum b_i + w_2|\mathcal{J}(S)|$ на каждой из итераций. Следующим фактором для оценки качества разработанных алгоритмов является вероятность построения хорошего по критерию $\Phi(S)$ расписания при каждой реализации алгоритма. Вычислительные результаты для десяти применений трех разработанных алгоритмов для одного и того же примера 1 из первого класса задач (примеры 1–5) представлены в табл. 3. Из этой таблицы следует, что дисперсия результатов в алгоритме имитации отжига ниже, чем для двух других разработанных алгоритмов. Например, если для пользователя является приемлемым получение значения первого слагаемого $w_1 \sum b_i$ целевой функции не менее 495, то применение алгоритма имитации отжига дает желаемый результат в 8 случаях из 10 реализаций алгоритма. Применение генетического алгоритма позволило бы достичь желаемого результата в 5 случаях из 10 реализаций алгоритма, а применение алгоритма поиска с запретами — в 4 случаях из 10 реализаций алгоритма. Таким образом, можно сделать вывод, что для решенных в эксперименте примеров средней размерности *вероятность достижения желаемого расписания с по-*

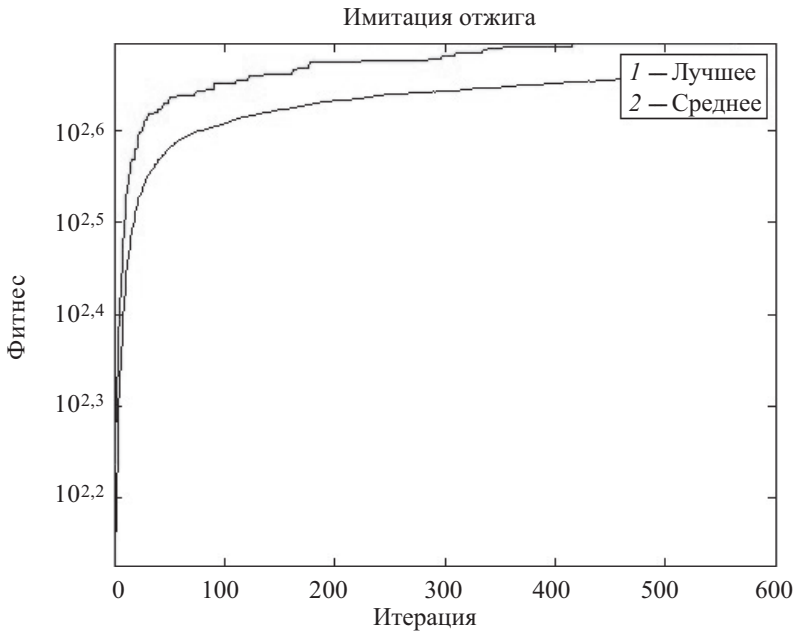

Рис. 3. Значения целевой функции после каждой итерации алгоритма имитации отжига.

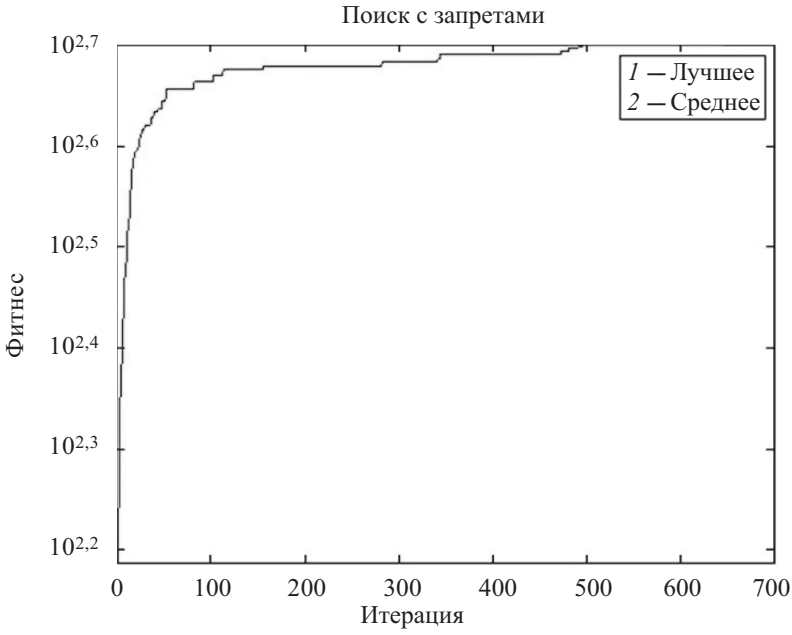

Рис. 4. Значения целевой функции после каждой итерации алгоритма поиска с запретами.

Таблица 3. Результаты вычислений, полученные в 10 случаях применения трех алгоритмов для решения примеров класса 1

Генетический алгоритм					Алгоритм имитации отжига				Поиск с запретами			
1	2	3	4	5	6	7	8	9	10	11	12	13
	$\sum b_i$	J_{ok}	CPU-time	n_{itr}	$\sum b_i$	J_{ok}	CPU-time	n_{itr}	$\sum b_i$	J_{ok}	CPU-time	n_{itr}
1	491	42	294	264	495	43	*1279*	614	502	45	370	648
2	492	42	307	262	488	42	*1162*	576	471	38	370	761
3	498	44	334	395	496	44	*1733*	543	491	42	213	421
4	487	42	404	347	472	46	*1421*	335	483	40	220	452
5	488	42	36	282	495	47	*2802*	548	500	45	317	627
6	495	43	298	250	502	45	*2572*	539	484	40	328	385
7	502	46	720	227	496	43	*2485*	638	506	47	884	1000
8	499	44	416	371	499	44	*4985*	661	495	43	552	388
9	507	47	315	279	495	43	*1543*	389	444	34	404	441
10	491	43	269	237	499	44	*2005*	500	491	42	669	462

Таблица 4. Результаты, полученные для 20 сгенерированных примеров с помощью генетического алгоритма ($n_{itr} = 1000$), алгоритма имитации отжига ($n_{itr} = 50$ и $NN = 100$) и алгоритма поиска с запретами ($NN = 1000$ и $lim_t = 50$)

Задачи	Генетический алгоритм				Алгоритм имитации отжига				Поиск с запретами			
	$\sum b_i$	J_{ok}	CPU-time	n_{itr}	$\sum b_i$	J_{ok}	CPU-time	n_{itr}	$\sum b_i$	J_{ok}	CPU-time	n_{itr}
1	507	47	315	279	502	47	2572	539	506	47	884	1000
2	554	48	657	295	546	45	1526	669	551	48	299	535
3	525	42	1010	362	525	44	2028	897	524	42	611	608
4	561	45	584	282	561	45	1716	831	565	45	297	523
5	557	47	351	292	560	47	2501	884	557	47	341	576
6	555	47	367	231	555	47	1415	633	557	44	477	890
7	501	45	475	258	505	46	1770	670	505	46	280	419
8	411	48	422	272	411	48	1465	682	**413**	**50**	525	767
9	**506**	**50**	359	282	**506**	**50**	1768	680	**506**	**50**	437	607
10	551	49	559	266	**552**	**50**	1547	571	551	49	221	454
11	526	47	590	288	529	47	1146	587	529	47	340	634
12	573	46	790	307	573	46	2781	699	572	46	401	698
13	512	48	761	337	513	49	1486	452	511	48	303	480
14	566	48	644	239	566	49	1969	510	566	49	432	671
15	485	44	655	260	487	45	1440	436	492	47	399	589
16	**540**	**50**	447	206	**540**	**50**	819	279	**540**	**50**	266	363
17	**515**	**50**	385	206	**515**	**50**	710	229	**515**	**50**	327	454
18	**510**	**50**	589	243	**510**	**50**	1249	291	**510**	**50**	242	405
19	**514**	**50**	601	262	**514**	**50**	1226	347	513	49	196	376
20	**564**	**50**	483	214	**564**	**50**	1264	373	**564**	**50**	207	393

Таблица 5. Сравнение полученных результатов для 10 примеров большой размерности

Задачи		Генетический алгоритм			Алгоритм имитации отжига			Поиск с запретами		
	$n \times m$	J_{ok}	CPU-time	$\sum b_i$	J_{ok}	CPU-time	$\sum b_i$	J_{ok}	CPU-time	$\sum b_i$
21	100×5	49	718	652	46	2852	639	41	356	571
22	100×10	78	743	872	85	4092	892	85	1265	903
23	200×5	51	1441	831	48	6873	789	44	2026	729
24	200×10	85	1538	1274	88	7453	1316	80	1511	1209
25	200×15	120	2180	1644	115	8582	1624	114	3428	1625
26	200×20	139	2141	1780	145	10375	1807	148	4036	1842
27	300×25	176	5147	2592	180	12964	2573	199	5211	2788
28	300×30	195	5325	2579	199	12430	2593	228	4470	2807
29	500×40	269	8114	4094	270	26368	4012	283	8739	4198
30	500×50	318	8402	4451	319	20601	4385	337	9231	4573

мощью алгоритма имитации отжига выше, чем при использовании двух других разработанных алгоритмов. Отметим, что сравнение алгоритмов было ограниченно рассмотренными в эксперименте примерами.

Второе отличие между алгоритмами — это время реализации алгоритма (CPU-time) на компьютере, необходимое для поиска достаточно хорошего расписания. Для примера 1 из первого класса все алгоритмы могут обеспечить получение прибыли в размере $\sum b_i \geqslant 502$. При этом оказалось, что $|\mathcal{J}(S)| \geqslant 45$ требований обслуживалось в установленные сроки при реализации наилучшего из построенных расписаний. С помощью генетического алгоритма такое расписание строится через 385 с, а с помощью алгоритма поиска с запретами — через 370 с, а при использовании алгоритма имитации отжига — через 2572 с. Таким образом, для построения расписания S, при котором $\sum b_i \geqslant 502$ и $|\mathcal{J}(S)| \geqslant 45$, алгоритм имитации отжига требует почти в 6 раз больше времени, чем любой из двух других алгоритмов. Отметим, что такие результаты были получены при построении 283 поколений генетическим алгоритмом, для 647 итераций алгоритма поиска с запретами и для 539 итераций алгоритма имитации отжига. В табл. 2 курсивом выделено наибольшее время работы алгоритмов (CPU-time) в секундах.

Аналогичная взаимосвязь по времени (CPU-time) работы алгоритмов наблюдалась и при решении задач средней размерности второго класса (примеры 6–10), третьего класса (примеры 11–15) и четвертого класса (примеры 16–20) (табл. 4), а также при решении всех примеров большой размерности 21–30 (табл. 5).

На основе предварительных вычислительных экспериментов параметры разработанных алгоритмов были изменены с целью повышения их эффективности и быстродействия. В табл. 4 представлены лучшие результаты, полученные в результате 10 реализаций алгоритмов для решения 20 задач средней размерности с помощью генетического алгоритма, алгоритма имитации отжига и поиска с запретами. Сравнивая результаты моделирования, полученные при использовании трех алгоритмов, можно заметить, что все алгоритмы

достаточно эффективны для нескольких из решенных примеров. Например, при решении примеров 1 и 2 с помощью генетического алгоритма получены наибольшие значения J_{ok}. Лучшие результаты были получены при решении примеров 5 и 13 с помощью алгоритма имитации отжига. Алгоритм поиска с запретами дал лучшие решения для примеров 4 и 15.

В табл. 2–4 подчеркнуты наилучшие из полученных значений $\sum b_i$ и $|\mathcal{J}(S)| = J_{ok}$. В табл. 4 жирным шрифтом выделены полученные наибольшие (оптимальные) значения $|\mathcal{J}(S)| = J_{ok} = |\mathcal{J}| = 50$. В силу теоремы 1 из равенства $|\mathcal{J}(S)| = |\mathcal{J}|$ следует, что и прибыль $\sum b_i$ также будет наибольшей (оптимальной) при расписании S. Оптимальные значения для прибыли $\sum b_i$ выделены жирным шрифтом в табл. 4.

При сравнении решений, полученных с помощью каждого из разработанных алгоритмов, было построено расписание с лучшим значением $\sum b_i$ и $|\mathcal{J}(S)| = J_{ok}$. Оптимальное значение $J_{ok} = 50$ было получено генетическим алгоритмом 6 раз (для примеров 9, 16–20), 7 раз (для примеров 9, 10, 16–20) при использовании алгоритма имитации отжига и 7 раз при использовании алгоритма поиска с запретами (для примеров 8, 9, 16–20). Из табл. 4 следует, что алгоритм имитации отжига построил расписание с лучшим значением $\sum b_i$ и (или) лучшим значением $|\mathcal{J}(S)| = J_{ok}$ 31 раз для примеров 1–20 средней размерности, алгоритм поиска с запретами — 27 раз и генетический алгоритм — 25 раз. Таким образом, можно сделать вывод, что для примеров средней размерности, которые были решены в вычислительном эксперименте, *алгоритм имитации отжига превосходит алгоритм поиска с запретами при нахождении лучших значений целевой функции, тогда как алгоритм поиска с запретами превосходит генетический алгоритм*. Все три разработанных алгоритма могут показать наилучшую эффективность при решении примеров средней размерности второго и четвертого классов.

5.3. Результаты вычислительного эксперимента для задач большой размерности

Разработанные алгоритмы были протестированы на примерах 21–30 для решения задач $Q/r_i, d_i/w_1 \sum b_i + w_2|\mathcal{J}(S)|$ большой размерности. Полученные результаты представлены в табл. 5. Для задач большой размерности 5 раз было построено лучшее расписание с помощью алгоритма поиска с запретами, 4 раза с помощью генетического алгоритма и один раз с помощью алгоритма имитации отжига (табл. 5). Было проведено сравнение, чтобы определить, сколько раз с помощью каждого разработанного алгоритма было построено расписание с лучшими значениями $\sum b_i$ и (или) $|\mathcal{J}(S)| = J_{ok}$ для примеров большой размерности 21–30. В табл. 5 подчеркнуты лучшие полученные значения $\sum b_i$ и $|\mathcal{J}(S)| = J_{ok}$. Из табл. 5 следует, что расписания, построенные с помощью алгоритма имитации отжига, имели наибольшее значение $\sum b_i$ и (или) $|\mathcal{J}(S)| = J_{ok}$ для трех примеров большой размерности 21–30, с помощью генетического алгоритма — для 6 примеров, с помощью алгоритма поиска с запретами — для 12 примеров. На основе решенных задач большой размерности можно заключить, что алгоритм поиска с запретами превосходит генетический алгоритм, а генетический алгоритм превосходит алгоритм имитации отжига.

С помощью разработанных алгоритмов не было найдено расписаний S с максимально возможными значениями $|\mathcal{J}(S)| = J_{ok} = n$ для примеров большой размерности 21–30 (см. табл. 5). К сожалению, пока нет точного алгоритма для нахождения расписания S с наибольшим (оптимальным) значением целевой функции $\Phi(S)$. Поэтому, если $|\mathcal{J}(S)| < |\mathcal{J}|$, нельзя оценить, как часто достигнутые значения целевой функции $\Phi(S)$, представленные в табл. 2–5, являются оптимальными значениями.

6. Практическое значение разработанных алгоритмов

На реально существующих предприятиях задача $Q/r_i, d_i/w_1 \sum b_i + +w_2|\mathcal{J}(S)|$ может возникать, когда необходимо получить максимально возможный доход и (или) увеличить количество запросов клиентов, которые реализует компания за приемлемое время (на горизонте планирования). Максимизация выручки может быть определена как распределение ресурсов (в данном случае ресурсами являются параллельные однотипные приборы) для реализации наиболее подходящих запросов клиентов на горизонте планирования для получения платежа (соответствующее требование должно быть своевременно обслужено для реализации запроса клиента). Если поток клиентских запросов превышает производственные возможности компании, некоторые запросы должны быть заранее отклонены как нерентабельные, поскольку любая компания имеет ограниченные ресурсы (обслуживающие приборы, станки, оборудование). Один из вопросов, который необходимо решить в первую очередь для компании, связан с определением соответствующих весов w_1 и w_2 целевой функции $\Phi(S) = w_1 \sum b_i + w_2|\mathcal{J}(S)|$ при условии, что выполняется равенство $w_1 + w_2 = 1$. Следует отметить, что увеличение первого слагаемого $(Obj1 = w_1 \sum b_i)$ целевой функции $\Phi(S) = w_1 \sum b_i + w_2|\mathcal{J}(S)| = Obj1 + Obj2$ означает, что компания предпочитает увеличить свой доход, в то время как количество клиентов компании на горизонте планирования может быть неизменно или даже уменьшено. С другой стороны, увеличение второго слагаемого $(Obj2 = w_2|\mathcal{J}(S)|)$ целевой функции Φ означает, что компания предпочитает увеличивать количество клиентов, тогда как выручка может быть сохранена или даже уменьшена на горизонте планирования.

Как показано в проведенных вычислительных экспериментах, если значения весов w_1 и w_2 близки друг к другу, то существует высокая корреляция между двумя слагаемыми $w_1 \sum b_i$ и $w_2|\mathcal{J}(S)|$ целевой функции $\Phi(S) = w_1 \sum b_i + w_2|\mathcal{J}(S)|$. В частности, при увеличении числа $|\mathcal{J}(S)|$ требований (запросов клиентов) $J_i \in \mathcal{J}(S)$, выполненных в заданные интервалы времени $[r_i, d_i]$ при расписании S, ожидается увеличение и прибыли компании. Аналогично, при увеличении прибыли компании увеличивается и число выполненных в срок требований $\mathcal{J}(S)$, если значения весов w_1 и w_2 достаточно близки один к другому. Однако если веса w_1 и w_2 различаются существенно, то часто можно отметить сравнительно низкую корреляцию между этими слагаемыми для оптимального расписания S.

Решая задачу $Q/r_i, d_i/w_1 \sum b_i + w_2|\mathcal{J}(S)|$ с различными множествами требований $\mathcal{J}$, можно строить различные множества требований $\mathcal{J}^*(S)$, которые соответствуют удовлетворенным запросам клиентов. Предлагается использо-

вать следующую итерационнную схему для получения более высокого дохода для компании и увеличения числа клиентов, чьи запросы будут удовлетворены компанией на горизонте планирования. У принимающих решения лиц есть возможность выбирать множество $\mathcal{J} = \{J_1, \ldots, J_n\}$ требований, которые включаются в подмножество множества $\mathcal{R} = \{\rho_1, \ldots, \rho_\lambda\}$ всех доступных запросов клиентов, где $\lambda \geqslant n$. Для каждого выбранного требования J_i условие теоремы 2 должно быть проверено во избежание рассмотрения запросов клиентов $\rho_i \in \mathcal{R}$, которые не могут быть реализованы в течение разрешенных интервалов времени $[r_i, d_i]$. Как было доказано в разделе 3.2, такой тест может быть реализован за время $O(n)$. Чтобы быстрее реализовать такие тесты, все требования, соответствующие множеству $\mathcal{R}$ доступных запросов клиентов, должны быть упорядочены в неубывающем порядке их разностей $d_i - r_i$.

После выбора множества требований $\mathcal{J}$, задача $Q/r_i, d_i/w_1 \sum b_i + +w_2|\mathcal{J}(S)|$ должна быть решена для множества $\mathcal{J}$ требований с использованием одного из разработанных алгоритмов. Если для построенного расписания S, выполняется равенство $\mathcal{J}(S) = \mathcal{J}$, то это расписание S является оптимальным для выбранного множества $\mathcal{J}$ требований (согласно теореме 1). Однако в построенном расписании S может быть простой приборов. В таком случае, если выполняется строгое неравенство $\lambda > n$, то существуют запросы клиентов, которые не принадлежат множеству $\mathcal{J}$. Лицо, принимающее решения, может увеличить множество $\mathcal{J} = \{J_1, \ldots, J_n\}$ путем выбора подходящих запросов клиентов из множества $\mathcal{R}$ и добавления соответствующих требований к множеству $\mathcal{J}$. Обозначим расширенное множество требований через $\mathcal{J}'$. Тогда задача $Q/r_i, d_i/w_1 \sum b_i + w_2|\mathcal{J}(S)|$ может быть снова решена для расширенного множества требований $\mathcal{J}'$ при условии, что $\mathcal{J} \subset \mathcal{J}'$. Если для построенного расписания S выполняется равенство $\mathcal{J}(S) = \mathcal{J}$, то расписание S является оптимальным для множества требований $\mathcal{J}'$.

Описанная итерационная схема может быть продолжена до тех пор, пока не будет выполняться неравенство $\mathcal{J}^*(S) < \mathcal{J}^*$. Полезно решать задачу $Q/r_i, d_i/w_1 \sum b_i + w_2|\mathcal{J}(S)|$ для последнего наибольшего множества $\mathcal{J}^*$ с использованием более эффективного алгоритма, чем тот, который использовался на предыдущих итерациях описанной схемы. С помощью более эффективного алгоритма можно найти и оптимальное расписание S. Однако процесс более точного решения задачи $Q/r_i, d_i/w_1 \sum b_i + w_2|\mathcal{J}(S)|$, как правило, требует больше времени. Описанная схема может использоваться и на более высоком уровне планирования, например, когда необходимо достичь наибольшего дохода и увеличить количество клиентских запросов, которые реализуются компанией, состоящей из m фабрик. В этом случае фабрика соответствует одному из параллельных однотипных приборов множества $\mathcal{M}$.

В рассмотренной задаче $Q/r_i, d_i/w_1 \sum b_i + w_2|\mathcal{J}(S)|$ все параметры компании предполагаются фиксированными. Однако в реальной компании могут существовать неопределенные параметры, которые трудно определить до решения задачи. Поэтому вышеприведенную итерационную схему необходимо расширить с помощью различных инструментов, чтобы учесть неопределенность максимизации доходов компании на основе более эффективного планирования.

7. Заключение

Исследована задача максимизации взвешенной суммы прибыли $w_1 \sum_{i \in \mathcal{J}(S)} b_i$ и числа требований $w_2|\mathcal{J}(S)|$, обслуженных в срок при расписании S. Требования из заданного множества $\mathcal{J} = \{J_1, \ldots, J_n\}$ должны быть обслужены на множестве параллельных однотипных приборов. $\mathcal{J}(S)$ обозначает подмножество требований $\mathcal{J}$, обслуженных в срок при расписании S. Для каждого требования $J_i \in \mathcal{J}$ заданы время готовности к обслуживанию $r_i > 0$ и директивный срок $d_i > r_i$ обслуживания требования. Каждое требование J_i должно быть обслужено на одном из m заданных однотипных параллельных приборов. Если требование J_i будет обслужено (т.е. будет начато и завершено) в течение интервала времени $[r_i, d_i]$, то прибыль b_i будет получена. В противном случае это требование будет удалено из расписания, а прибыль b_i получена не будет.

Разработаны генетический алгоритм, алгоритм имитации отжига и алгоритм поиска с запретами для приближенного решения рассматриваемой задачи. Исследовано влияние различных параметров этих алгоритмов, например количество поколений (итераций), количество сгенерированных соседей за одну итерацию, размер популяции для генетического алгоритма, время работы алгоритма. Было проведено сравнение разработанных алгоритмов по значению целевой функции $\Phi(S) = w_1 \sum b_i + w_2|\mathcal{J}(S)|$ и затраченного времени процессора для случайно сгенерированных задач средней и большой размерности.

В дальнейших исследованиях было бы полезно найти верхнюю границу значений целевой функции $\Phi(S) = w_1 \sum b_i + w_2|\mathcal{J}(S)|$. Интересная область исследований связана с различными критериями оптимальности планирования множества операций на параллельных однотипных приборах. Представляет интерес разработка точного алгоритма для решения задачи $Q/r_i, d_i/w_1 \sum b_i + w_2|\mathcal{J}(S)|$ средней размерности. Такой алгоритм позволит сравнивать оптимальные расписания с расписаниями, полученными эвристическими алгоритмами. Поскольку большинство практических задач планирования имеют неопределенные параметры, было бы полезно исследовать задачу $Q/r_i, d_i/w_1 \sum b_i + w_2|\mathcal{J}(S)|$ с неопределенными длительностями операций аналогично тому, как это было реализовано в [25] для планирования множества операций на одном приборе.

СПИСОК ЛИТЕРАТУРЫ

1. *Brucker P., Sotskov Y.N., Werner F.* Complexity of shop-scheduling problem with fixed number of jobs: a survey // Math. Methods Oper. Res. 2007. V. 65. No. 3. P. 461–481.
2. *Graham R.E., Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G.* Optimization and approximation in deterministic sequencing and scheduling: a survey // Ann. Discret. Math. 1979. V. 4. P. 287–326.
3. *Anglani A., Grieco A., Guerriero E., Musmanno R.* Robust scheduling of parallel machines with sequence-dependent set-up cost // Eur. J. Oper. Res. 2005. V. 161. No. 3. P. 704–720.
4. *Feng G., Lau H.C.* Efficient algorithm for machine scheduling problems with earliness and tardiness penalties // Ann. Oper. Res. 2008. V. 159. No. 1. P. 83–95.

5. *Berrichi A., Yalaoui F.* Efficient bi-objective ant colony approach to minimize total tardiness and system unavailability for a parallel machine scheduling problem // Int. J. Advanced Manufact. Technol. 2013. V. 68. No. 9–12. P. 2295–2310.
6. *Lin Y.K., Lin C.W.* Dispatching rules for unrelated parallel machine scheduling with release dates // Int. J. Advanced Manufact. Technol. 2013. V. 67. No. 1–4. P. 269–279.
7. *Balin S.* Non-identical parallel machine scheduling using genetic algorithm // Expert Syst. Appl. 2011. V. 38. No. 6. P. 6814–6821.
8. *Juraszek J., Sterna M., Pesch E.* Revenue maximization on parallel machines, Institute of Computing Science, Poznan Univer. Technol. Piotrowo 2, Poznan, Poland. P. 960–965.
9. *Balin S.* Non-identical parallel machine scheduling with fuzzy processing times using robust genetic algorithm and simulation // Int. J. Innovat. Comput. Inform. Control. 2012. V. 8. No. 1-B. P. 727–745.
10. *Li K., Yang S.-L.* Non-identical parallel-machine scheduling research with minimizing total weighted completion times: models, relaxations and algorithms // Appl. Math. Modell. 2009. V. 33. No. 4. P. 2145–2158.
11. *Xu S., Bean J.C.* A genetic algorithm for scheduling parallel non-identical batch processing machines // Comput. Intelligen. Scheduling, SCIS '07. IEEE Sympos. 2007. P. 143–150.
12. *Rodriguez F.J., Blum C., Garcia-Martinez C., Lozano M.* GRASP with path-relinking for the non-identical parallel machine scheduling problem with minimising total weighted completion times // Ann. Oper. Res. 2012. V. 201. No. 1. P. 383–401.
13. *Supithak W., Plongon K.* Memetic algorithm for non-identical parallel machines scheduling problem with earliness and tardiness penalties // Int. J. Manufactur. Technol. Management. 2011. V. 22. No. 1. P. 26–38.
14. *Logendran R., McDonell B., Smuckera B.* Scheduling unrelated parallel machines with sequence-dependent setups // Comput. Oper. Res. 2007. V. 34. No. 11. P. 3420–3438.
15. *Anagnostopoulos G., Rabadi G.* A simulated annealing algorithm for the unrelated parallel machine scheduling problem // Autom. Congr. 2002. Proc. 5 Biannual World. 2002. V. 14. P. 115–120.
16. *Sivrikaya-Serifoglu F., Ulusoy G.* Parallel machine scheduling with earliness and tardiness penalties // Comput. Oper. Res. 1999. V. 26. No. 8. P. 773–787.
17. *Bilge U., Kirac F., Kurtulan M., Pekgun P.* A tabu search algorithm for parallel machine total tardiness problem // Comput. Oper. Res. 2004. V. 31. No. 3. P. 397–414.
18. *Tavakkoli-Moghaddam R., Taheri F., Bazzazi M.* Multi-objective unrelated machines scheduling with sequence-dependent setup times and precedence constrains // IEE Transactions A: Basics, 2008. V. 21. No. 3. P. 269–278.
19. *Dunstall S., Wirth A.* Heuristic methods for the identical parallel machine problem with set-up times // Comput. Oper. Res. 2005. V. 32. No. 9. P. 2479–2491.
20. *Sterna M., Juraszek J., Pesch E.* Revenue maximization on parallel machines, Book Chapter // Oper. Res. Proc. 2008. P. 153–158.
21. *Islam M., Khanna G., Sadayappan P.* Revenue maximization in market-based parallel job schedulers, Technical Report, Ohio State University, OSU-CISRC-4/08-TR16, Ohio, USA, 2008, 1–13.
22. *Dawande M., Drobouchevitch I., Rajapakshe T., Sriskandarajah C.* Analysis of revenue maximazation under two movie-screening policies // Product. Oper. Management. 2010. V. 19. No. 1. P. 111–124.

23. *Feng G., Garg S., Buyya R., Li W.* Revenue maximization using adaptive resource provisioning in cloud computing environments, 2012 ASM/IEEE 13th Int. Conf. Grid Comput. // IEEE Comput. Soc. 2012. P. 192–200. DOI 10.1109/Grid.2012.16
24. *Glover F.* Future paths for integer programming and links to artificial intelligence // Comput. Oper. Res. 1986. V. 13. No. 5. P. 533–549.
25. *Sotskov Y.N., Lai T.-C.* Minimizing total weighted flow time under uncertainty using dominance and a stability box // Comput. Oper. Res. 2012. V. 39. No. 6. P. 1271–1289.
26. *Carter M.W., Price C.C.* Operations research: A practical introduction, Textbook, CRC Press, Boca Raton, 2001.

Статья представлена к публикации членом редколлегии А.А. Лазаревым.

Поступила в редакцию 20.03.2018

После доработки 25.09.2018

Принята к публикации 08.11.2018