ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ
Сентябрь—октябрь 2011. Том 18, № 5. С. 54–79

УДК 519.7

О ЗАДАЧЕ СОСТАВЛЕНИЯ РАСПИСАНИЙ С ГРУППИРОВКОЙ МАШИН ПО ТЕХНОЛОГИЯМ

А. В. Еремеев, Ю. В. Коваленко

Аннотация. Рассматривается задача составления расписаний многопродуктового производства. Особенностью постановки является то, что каждый продукт имеет несколько технологий производства, при выполнении которых используется сразу несколько машин, работающих одновременно. Задача исследуется в двух вариантах: с возможностью прерываний выполнения технологий и без неё. Для обоих случаев построены модели частично целочисленного линейного программирования и разработаны генетические алгоритмы с равномерным кроссинговером и с оптимальной рекомбинацией. Исследована сходимость предложенных алгоритмов. Выполнены численные эксперименты, проведён анализ сложности задачи.

Ключевые слова: расписание, переналадка, технология, целочисленное линейное программирование, генетический алгоритм.

Введение

На практике достаточно часто возникают задачи составления расписаний для производства, где рассматриваются вещества, операции и машины. Вещества подразделяются на исходное сырьё, промежуточные и окончательные продукты. В результате выполнения операций происходит преобразование одних веществ в другие. Кроме того, необходимо учитывать погрузку, хранение и транспортировку продуктов, переналадку машин и т. д. Продукты могут производиться либо непрерывно, либо партиями. Требуется построить такое допустимое расписание выпуска продукции, при котором минимизируется общее время завершения выполнения операций.

Ряд работ (см., например, [14, 16, 20, 23, 25]) отличаются тем, что в них рассматриваются не отдельные операции, а технологии, представляющие собой набор операций, в результате действия которых может быть получен тот или иной продукт. Для выполнения технологии необходимо наличие некоторой совокупности машин, работающих одновременно или

последовательно друг за другом. В свою очередь каждый продукт может быть произведён одной или более технологиями.

В настоящей статье рассматривается задача составления расписаний с группировкой машин по технологиям, возникающая, например, при производстве пластмасс и являющаяся частным случаем задачи, предложенной в [16]. Здесь в качестве операций рассматриваются химические реакции. Отличительной особенностью данной задачи является то, что при выполнении технологии используется сразу несколько машин, работающих одновременно, при этом если машина переключается с одной технологии на другую, то необходимо выполнять переналадку. В соответствии с известными обозначениями [10, 13, 23] исследуемая задача обозначается через $P|\mathrm{set}_i,\ \mathrm{pmtn},\ \mathrm{s}_{luq}|C_{\max}$, если допускается прерывание выполнения технологий, и $P|\mathrm{set}_i,\ \mathrm{s}_{luq}|C_{\max}$ — в противном случае.

В задачах составления расписаний [1, 13] рассматриваются технологии, задействующие при своем выполнении только одну машину. В [10, 11, 14, 20, 23, 24] проводится анализ сложности задач составления расписаний с группировкой машин по технологиям, в которых длительности переналадок машин равны нулю. В [16] технологии рассматриваются при обсуждении постановки задачи, однако в модели они в явном виде не представлены. Технологии учитываются в явном виде в предложенной нами модели частично целочисленного линейного программирования [6] для задачи $P|\mathrm{set}_i,\ \mathrm{pmtn},\ \mathrm{s}_{luq}|C_{\max}$. Данная модель имеет меньшее число переменных и ограничений, чем модель [16] для случая задачи $P|\mathrm{set}_i,\ \mathrm{pmtn},\ \mathrm{s}_{luq}|C_{\max}$. Моделирование технологий в явном виде позволяет применить для приближённого решения задач большой размерности эволюционные алгоритмы, где кодировка решений основана на выборе совокупности технологий.

В статье для рассматриваемых задач предложены генетический алгоритм с равномерным кроссинговером и генетический алгоритм с оптимальной рекомбинацией. В процессе их выполнения решается серия подзадач пониженной размерности в соответствии с предложенной нами моделью частично целочисленного линейного программирования. Исследована сходимость разработанных генетических алгоритмов. Кроме того, выделены новые NP-трудные в сильном смысле и полиномиально разрешимые частные случаи рассматриваемых задач.

Статья построена следующим образом. В разд. 1 приводится постановка задачи, исследуется её сложность и формулируется модель частично целочисленного линейного программирования для случая, когда длительности переналадки удовлетворяют неравенству треугольни-

ка. В разд. 2 излагается общая схема генетического алгоритма со стационарной схемой воспроизведения и вариант его адаптации к исследуемой задаче в случае равномерного кроссинговера. В разд. 3 предлагается генетический алгоритм с оптимальной рекомбинацией. В разд. 4 приводятся результаты численных экспериментов. В разд. 5 обсуждаются основные результаты работы.

1. Постановка задачи и анализ её сложности

Рассмотрим предприятие, выпускающее k различных продуктов. Обозначим через $V_i \in \mathbb{R}^+$ требуемый объём производства продукта $i = 1, \ldots, k$. Здесь и далее $\mathbb{R}^+$ — множество положительных вещественных чисел. Пусть m — число машин, которые могут использоваться при выпуске продукции.

Для каждого продукта $i = 1, \ldots, k$ указана одна или более технологий его производства. Пусть U — множество технологий, где $|U| = d$, каждая из которых характеризуется набором одновременно занимаемых машин $M_u \subseteq \{1, \ldots, m\}$, $u \in U$, т. е. если производится продукт i по технологии u, то одновременно задействованы все машины, относящиеся к данной технологии. В любой момент каждая машина не может быть задействована более чем в одной технологии.

Пусть $U_i \subseteq U$ — множество технологий по производству продукта i, $i = 1, \ldots, k$, для каждой из которых задан объём $a_u \in \mathbb{R}^+$ выпуска данного продукта в единицу времени, $u \in U_i$. Предполагается, что для выпуска продукта i может быть использовано несколько технологий из множества U_i, $i = 1, \ldots, k$.

Для машины l заданы длительности переналадки этой машины с технологии u на технологию q, обозначенные через $s_{luq} \in \mathbb{R}_+$ (здесь $\mathbb{R}_+$ — множество неотрицательных вещественных чисел), для всех u, $q \in K_l$, где $K_l = \{u \in U : \ l \in M_u\}$ — множество технологий, использующих машину $l = 1, \ldots, m$.

Для каждого продукта $i = 1, \ldots, k$ необходимо определить, какие технологии $u \in U_i$ будут использоваться для его производства, и для выбранных технологий составить расписание их выполнения с учётом переналадок машин и невозможности одновременного использования одной машины в различных технологиях таким образом, чтобы общее время окончания производства всех продуктов $C_{\max}$ в объёмах $V_1, \ldots, V_k$ было минимально. Задача рассматривается в двух вариантах: с возможностью прерываний выполнения технологий $(P|\mathrm{set}_i,\ \mathrm{pmtn},\ \mathrm{s}_{luq}|C_{\max})$ и без неё $(P|\mathrm{set}_i,\ \mathrm{s}_{luq}|C_{\max})$.

На практике достаточно часто встречаются задачи составления производственных расписаний, где длительности переналадки удовлетворяют неравенству треугольника:

$$s_{luq} + s_{lqp} \geqslant s_{lup}, \quad l = 1, \ldots, m,\ u, q, p \in K_l. \tag{1}$$

Обозначим этот частный случай через $P|\mathrm{set}_i,\ \mathrm{pmtn},\ \Delta\mathrm{s}_{luq}|C_{\max}$, если допускается прерывание выполнения технологий, и $P|\mathrm{set}_i,\ \Delta\mathrm{s}_{luq}|C_{\max}$, если не допускается. Указанные задачи являются NP-трудными в сильном смысле, так как в частном случае при $m = 1$ к ним сводится полуметрическая задача о кратчайшем гамильтоновом пути, являющаяся NP-трудной в сильном смысле [22].

Труднорешаемость задач при нулевых длительностях переналадки. В [20, 23] показано, что при нулевых длительностях переналадки исследуемые задачи являются труднорешаемыми. Эти результаты получены с использованием задач о минимальной вершинной раскраске графа и минимальной дробной вершинной раскраске графа. Обе задачи задаются графом $G = (V, E)$, где $V = \{v_1, \ldots, v_{|V|}\}$ — множество вершин, а $E = \{e_1, \ldots, e_{|E|}\}$ — множество рёбер. В первой задаче требуется найти хроматическое число $\chi(G)$ графа G, а во второй — дробное хроматическое число $\chi^*(G)$ графа G, т. е. минимальное значение $\lambda_1 + \cdots + \lambda_z$, где $z \in \mathbb{Z}^+$ (здесь $\mathbb{Z}^+$ — множество положительных целых чисел), $\lambda_j \in \mathbb{R}^+$ для всех $j = 1, \ldots, z$, такое, что найдётся набор $\mathcal{W} = \{W_1, \ldots, W_z\}$ независимых множеств вершин графа G, для которого имеют место неравенства

$$\sum_{W_j \in \mathcal{W}:\ v \in W_j} \lambda_j \geqslant 1, \quad v \in V.$$

Задачи о минимальной вершинной раскраске и минимальной дробной вершинной раскраске графа являются NP-трудными в сильном смысле [3, 17], и если справедлива известная гипотеза о не включении классов NP$\not\subseteq$ZPP [26], то при любом $\varepsilon > 0$ для них не существует полиномиальных приближённых алгоритмов с точностью аппроксимации $|V|^{1-\varepsilon}$ [15, 23].

Задача о минимальной вершинной раскраске графа сводится как частный случай к задаче $P|\mathrm{set}_i,\ \mathrm{s}_{luq} = 0|C_{\max}$ [20], если положить $k = |V|$, $m = |E|$, $V_i = |U_i| = 1$, $i = 1, \ldots, k$, производительность a_u равной 1, $u \in U_i$, $i = 1, \ldots, k$; ребру e_l графа G, $l = 1, \ldots, m$, поставить в соответствие машину l, а вершине $v_i \in V$, $i = 1, \ldots, k$, — продукт i и набор одновременно занимаемых машин $M_{u_i} = \{l : v_i \in e_l,\ l = 1, \ldots, m\}$, где u_i

такое, что $U_i = \{u_i\}$. Аналогичным образом задача о минимальной дробной вершинной раскраске графа сводится как частный случай к задаче $P|\text{set}_i,\ \text{pmtn},\ \text{s}_{luq} = 0|C_{\max}$ [23]. Поэтому из результатов [15, 20, 23] непосредственно следует

Утверждение 1. *Задачи* $P|\text{set}_i,\ \text{pmtn},\ \text{s}_{luq} = 0|C_{\max}$ *и* $P|\text{set}_i,\ \text{s}_{luq} = 0|C_{\max}$ *даже в частном случае, когда для всякого продукта имеется только одна технология его производства, а каждая машина пригодна для работы только по двум технологиям, являются* NP*-трудными в сильном смысле, и если справедлива гипотеза* NP$\not\subseteq$ZPP*, то при любом* $\varepsilon > 0$ *для их решения не существует полиномиального приближённого алгоритма с точностью аппроксимации* $k^{1-\varepsilon}$.

Покажем, что в случае, когда требуется выпустить только один продукт, но с использованием произвольного числа технологий, задачи также являются труднорешаемыми.

Утверждение 2. (i) *Задачи* $P|\text{set}_i,\ \text{pmtn},\ \text{s}_{luq} = 0|C_{\max}$ *и* $P|\text{set}_i,\ \text{s}_{luq} = 0|C_{\max}$ *являются* NP*-трудными в сильном смысле даже в частном случае, когда требуется произвести один продукт с использованием технологий одинаковой производительности, задействующих при своём выполнении не более трёх машин.*

(ii) *Если справедлива гипотеза* NP$\not\subseteq$ZPP*, то для любого* $\varepsilon > 0$ *не существует полиномиального приближённого алгоритма с точностью аппроксимации* $d^{1-\varepsilon}$ *для решения задач* $P|\text{set}_i,\ \text{pmtn},\ \text{s}_{luq} = 0|C_{\max}$ *и* $P|\text{set}_i,\ \text{s}_{luq} = 0|C_{\max}$*, даже если число продуктов* k *равно* 1.

Доказательство. (i) Покажем, что к рассматриваемым задачам сводится задача об упаковке множества: задано конечное множество $\mathcal{X} = \{x_1, \ldots, x_{|\mathcal{X}|}\}$ и набор $\mathcal{S} = \{S_1, \ldots, S_{|\mathcal{S}|}\}$ подмножеств множества $\mathcal{X}$, требуется найти семейство непересекающихся подмножеств из набора $\mathcal{S}$ максимальной мощности ω^*.

Построим индивидуальную задачу составления расписаний $P|\text{set}_i,\ \text{s}_{luq} = 0|C_{\max}$ ($P|\text{set}_i,\ \text{pmtn},\ \text{s}_{luq} = 0|C_{\max}$). Пусть число продуктов k равно 1, число машин m равно $|\mathcal{X}|$, множество технологий $U = \{1, \ldots, |\mathcal{S}|\}$. Подмножеству S_u, $u = 1, \ldots, |\mathcal{S}|$, поставим в соответствие технологию u и набор одновременно занимаемых машин $M_u = \{l :\ x_l \in S_u, l = 1, \ldots, m\}$. Положим требуемый объём V_1 равным 1, и $a_u = 1, u \in U$.

Заметим, что при сделанных предположениях достаточно рассматривать только те допустимые расписания, в которых на каждой машине выполняется не более одной технологии, причём без прерываний, и все задействованные машины работают равное время. Действительно, в лю-

бом оптимальном расписании может быть найден момент времени, когда число одновременно выполняемых технологий максимально. Рассмотрим расписание, состоящее в одновременном использовании указанного набора технологий до момента выпуска продукта в требуемом объёме. Данное расписание также будет оптимальным.

Таким образом, существует взаимно однозначное соответствие между упаковками из ω подмножеств, $\omega = 1, 2, \ldots, \omega^*$, и допустимыми расписаниями указанного вида длины $1/\omega$. Тогда из NP-трудности в сильном смысле [3] задачи об упаковке множества даже при $|S_u| \leqslant 3$, $u = 1, \ldots, |\mathcal{S}|$, следует (i).

(ii) Если справедлива гипотеза NP$\not\subseteq$ZPP, то, как следует из результата [18], для любого $\varepsilon > 0$ не существует полиномиального приближённого алгоритма решения задачи об упаковке множества с точностью аппроксимации $|\mathcal{S}|^{1-\varepsilon}$. Тогда из приведённой сводимости следует требуемое. Утверждение 2 доказано.

В [11, 20, 23, 24] могут быть найдены другие NP-трудные частные случаи задач $P|\mathrm{set}_i,\ \mathrm{s}_{luq} = 0|C_{\max}$ и $P|\mathrm{set}_i,\ \mathrm{pmtn},\ \mathrm{s}_{luq} = 0\ |C_{\max}$.

Модель частично целочисленного линейного программирования. Учитывая структуру рассматриваемой задачи, можно предложить модель частично целочисленного линейного программирования, основанную на тех же принципах, что и в [1, 16, 21].

Перед описанием модели определим понятие точки событий, аналогичное введённому в [21]. *Точка событий* — это группа переменных задачи, задающих некоторый набор технологий и моменты начала и окончания выполнения технологий из этого набора. В одной точке событий каждая машина может быть задействована не более чем в одной технологии. Множество точек событий обозначим через $N = \{1, \ldots, n_{\max}\}$, где параметр $n_{\max}$ выбирается достаточно большим на основе априорных оценок или вычислительных экспериментов.

Основными переменными для построения модели будут бинарные переменные w_{un} такие, что $w_{un} = 1$, если технология u выполняется в точке событий n, иначе $w_{un} = 0$. Кроме того, введём вещественные переменные, которые в случае, если технология u выполняется в точке событий n, имеют следующий смысл: T^s_{un} — время начала выполнения технологии u в точке событий n, T^f_{un} — время завершения выполнения технологии u в точке событий n. Переменная $C_{\max}$ соответствует моменту завершения производства всех продуктов.

Введём следующие обозначения:

I — множество продуктов, $|I| = k$; M — множество машин, $|M| = m$;

$H = \sum\limits_{i \in I} \max\limits_{u \in U_i} V_i/a_u + (k-1) \max\limits_{l \in M,\ u,q \in K_l} s_{luq}$ — оценка сверху длины расписания. Времени H заведомо достаточно для производства всех продуктов.

Тогда модель частично целочисленного линейного программирования для задачи $P|\mathrm{set}_i,\ \mathrm{pmtn},\ \Delta\mathrm{s}_{luq}|C_{\max}$ может быть записана следующим образом:

$$C_{\max} \to \min, \tag{2}$$

$$T^f_{un} \leqslant C_{\max}, \quad u \in U,\ n \in N, \tag{3}$$

$$\sum_{u \in K_l} w_{un} \leqslant 1, \quad l \in M,\ n \in N, \tag{4}$$

$$T^s_{u,n+1} \geqslant T^f_{un}, \quad u \in U,\ n \in N,\ n \neq n_{\max}, \tag{5}$$

$$T^s_{u,n+1} \geqslant T^f_{qn} + s_{lqu} \cdot w_{u,n+1} - H \cdot (1 - w_{u,n+1}), \tag{6}$$

$$l \in M,\ u,q \in K_l,\ u \neq q,\ n \in N,\ n \neq n_{\max},$$

$$T^s_{un} \geqslant -H \cdot (1 - w_{un}), \quad u \in U,\ n \in N, \tag{7}$$

$$T^f_{un} \geqslant T^s_{un}, \quad u \in U,\ n \in N, \tag{8}$$

$$T^f_{un} - T^s_{un} \leqslant w_{un} \cdot \max_{q \in U_i} V_i/a_q, \quad i \in I,\ u \in U_i,\ n \in N, \tag{9}$$

$$\sum_{n \in N} \sum_{u \in U_i} a_u \cdot \left(T^f_{un} - T^s_{un}\right) \geqslant V_i, \quad i \in I, \tag{10}$$

$$w_{un} \in \{0,1\}, \quad u \in U,\ n \in N. \tag{11}$$

Целевая функция (2) и неравенство (3) задают критерий минимизации момента окончания производства всех продуктов. Ограничение (4) выражает тот факт, что в каждой точке событий машина l используется не более чем в одной технологии, ограничение (5) — что время начала выполнения технологии u в точке событий $n+1$ должно быть не меньше, чем время окончания её выполнения в точке событий n. Условие (6) говорит о том, что время начала технологии u на машине l должно быть не меньше, чем время окончания предыдущей технологии на той же машине плюс длительность переналадки. Однако, если технология u является первой на машине l, то длительность переналадки на неё учитывать не нужно. Это обеспечивается благодаря тому, что переменные T^s_{qn} во всех предшествующих точках событий могут принимать отрицательные значения. Если же технология u имеет место в точке событий n, т. е. $w_{un} = 1$,

то время начала её использования должно быть неотрицательным, что обеспечивается условием (7). Неравество (8) гарантирует неотрицательность длительности технологий. Если технология u в точке событий n не выполняется, т. е. $w_{un} = 0$, то её длительность должна быть равна нулю, что обеспечивается условием (9). Ограничение (10) гарантирует выпуск продукции в заданном объёме.

Необходимо отметить, что условие (6) будет гарантировать получение оптимального решения задачи $P|\mathrm{set}_i,\ \mathrm{pmtn},\ \mathrm{s}_{luq}|C_{\max}$, только если длительности переналадки удовлетворяют неравенству треугольника.

Модель частично целочисленного линейного программирования для задачи $P|\mathrm{set}_i,\ \Delta\mathrm{s}_{luq}|C_{\max}$ может быть получена путём добавления к ограничениям (2)–(11) неравенства

$$\sum_{n\in N} w_{un} \leqslant 1, \quad u \in U, \tag{12}$$

гарантирующего непрерывность выполнения каждой технологии.

В [6] предложена модель частично целочисленного линейного программирования для случая произвольных длительностей переналадки и возможности прерываний выполнения технологий. Однако результаты вычислительного эксперимента [6] показали, что при решении задачи $P|\mathrm{set}_i, \mathrm{pmtn}, \Delta\mathrm{s}_{luq}|C_{\max}$ предпочтительнее использовать модель (2)–(11). Это свойство имеет место и для задачи $P|\mathrm{set}_i,\ \mathrm{s}_{luq}|C_{\max}$, поскольку модель данной задачи может быть получена добавлением к модели из [6] неравенства (12).

Полиномиально разрешимые частные случаи. В модели (2)–(12) для получения оптимального решения задачи $P|\mathrm{set}_i,\ \Delta\mathrm{s}_{luq}|C_{\max}$ достаточно положить $n_{\max} = d$, поскольку прерывание выполнения технологий не допускается. Обозначим через $\mathcal{P}_{ЛП}$ задачу линейного программирования, получаемую в результате фиксации значений всех булевых переменных (w_{un}) в модели (2)–(12). Задача $\mathcal{P}_{ЛП}$ при $n_{\max} = d$ имеет полиномиально ограниченное число переменных и, следовательно, является полиномиально разрешимой (см., например, [9]). Пусть $\tau_{ЛП}$ — оценка сверху временно́й сложности алгоритма решения задачи $\mathcal{P}_{ЛП}$.

Задачу $P|\mathrm{set}_i,\ \Delta\mathrm{s}_{luq}|C_{\max}$, в которой число технологий ограничено константой, обозначим через $Pd|\mathrm{set}_i,\ \Delta\mathrm{s}_{luq}|C_{\max}$. Данная задача может быть сведена к решению $(n_{\max}+1)^d$ задач $\mathcal{P}_{ЛП}$, где $n_{\max} = d$. Следовательно, имеет место

Утверждение 3. *Задача $Pd|\mathrm{set}_i,\ \Delta\mathrm{s}_{luq}|C_{\max}$ является полиномиально разрешимой за время $O(\tau_{ЛП} d^d)$.*

В модели (2)–(11) достаточно положить $n_{\max} = d^m$ для получения оптимального решения задачи $P|\mathrm{set}_i,\ \mathrm{pmtn},\ \Delta\mathrm{s}_{luq}|C_{\max}$. Действительно, количество различных наборов технологий, которые могут выполняться одновременно, не превосходит $\prod_{l=1}^{m} f_l \leqslant d^m$, где $f_l = |K_l| + 1$, если $|K_l| < d$, и $f_l = d$ в противном случае. Кроме того, для задачи $P|\mathrm{set}_i,\ \mathrm{pmtn},\ \Delta\mathrm{s}_{luq}|C_{\max}$ найдётся оптимальное расписание, использующее каждый такой набор не более одного раза, поскольку ввиду неравенства (1) в данном случае применим перестановочный приём (см., например, [8]).

Обозначим через $\mathcal{P}'_{ЛП}$ задачу линейного программирования, получаемую в результате фиксации значений всех булевых переменных (w_{un}) в модели (2)–(11). Задача $\mathcal{P}'_{ЛП}$ при $n_{\max} = d^m$ и числе машин, ограниченном константой, является полиномиально разрешимой. Пусть $\tau'_{ЛП}$ — оценка сверху временно́й сложности алгоритма решения задачи $\mathcal{P}'_{ЛП}$. Задачу $P|\mathrm{set}_i, \mathrm{pmtn}, \Delta\mathrm{s}_{luq}|C_{\max}$, в которой число машин и продуктов ограничено константой, обозначим через $Pmk|\mathrm{set}_i,\ \mathrm{pmtn},\ \Delta\mathrm{s}_{luq}|C_{\max}$. Данная задача может быть сведена к решению $2^{dn_{\max}}$ задач $\mathcal{P}'_{ЛП}$, где $n_{\max} = d^m$. Поскольку общее число технологий d не превосходит $k(2^m - 1)$, имеет место

Утверждение 4. *Задача* $Pmk|\mathrm{set}_i,\ \mathrm{pmtn},\ \Delta\mathrm{s}_{luq}|C_{\max}$ *является полиномиально разрешимой за время* $O(\tau'_{ЛП} 2^{(k(2^m-1))^{m+1}})$.

Различные полиномиально разрешимые частные случаи задач $P|\mathrm{set}_i, \mathrm{s}_{luq} = 0|C_{\max}$ и $P|\mathrm{set}_i, \mathrm{pmtn}, \mathrm{s}_{luq} = 0|C_{\max}$ можно найти в [10, 20, 23, 24].

2. Генетический алгоритм с равномерным кроссинговером

Ввиду утверждений 1 и 2 эффективный поиск приближённого решения исследуемых задач с какой-либо практически приемлемой гарантированной оценкой точности не представляется возможным, поэтому имеет смысл разработка метаэвристик, в частности, генетических алгоритмов (ГА). В этом разделе предлагается ГА для задачи $P|\mathrm{set}_i, \Delta\mathrm{s}_{luq}|C_{\max}$. Поскольку всякое допустимое её решение является допустимым решением задачи $P|\mathrm{set}_i, \mathrm{pmtn}, \Delta\mathrm{s}_{luq}|C_{\max}$, данный алгоритм может быть также применён к последней.

ГА является алгоритмом случайного поиска, в котором моделируется процесс развития популяции особей [19]. Каждая особь соответствует некоторому решению задачи (фенотипу), которое представлено в алгоритме как некоторая строка символов (генотип). Под значением целевой функции генотипа будем понимать значение целевой функции фе-

нотипа, ему соответствующего. При этом в случае задачи минимизации чем меньше значение целевой функции генотипа, тем больше он имеет шансов быть выбранным в качестве родительского для построения новых особей. Особи-потомки строятся на основе имеющихся особей путём применения вероятностных операторов кроссинговера (рекомбинации) и мутации.

Обозначим через $\Pi^t = (\zeta^{1,t}, \ldots, \zeta^{r,t})$ популяцию поколения t, $t = 0, 1, 2, \ldots$, в ГА, где r — численность популяции, которая является фиксированной от начала и до конца работы алгоритма. ГА со стационарной схемой воспроизведения выглядит следующим образом [27].

ГА со стационарной схемой воспроизведения

Шаг 0. Положить $t := 0$.

Шаг 1. Случайным образом построить начальную популяцию Π^0.

Шаг 2. Пока не выполнен критерий остановки, выполнять:

2.1. Выбрать из популяции Π^t два родительских генотипа p^1, p^2.

2.2. Применить к p^1 и p^2 оператор мутации.

2.3. Построить двух потомков c^1 и c^2, применяя оператор кроссинговера.

2.4. Удалить из популяции Π^t два генотипа, выбранных с равномерным распределением среди имеющих значение целевой функции выше среднего в популяции Π^t, и поместить на их место c^1 и c^2. Положить $\Pi^{t+1} := \Pi^t$.

2.5. Положить $t := t + 1$.

Шаг 3. Результатом работы алгоритма является фенотип лучшего из найденных генотипов.

Предлагаемый ГА со стационарной схемой воспроизведения для решения задачи $P|\mathrm{set}_i,\ \Delta\mathrm{s}_{luq}|C_{\max}$ основывается на решении серии подзадач пониженной размерности в соответствии с моделью (2)–(12). Определим схему представления решений, операторы селекции, кроссинговера и мутации.

Схема представления решений. Перенумеруем элементы из множества технологий U_i производства продукта $i = 1, \ldots, k$, обозначая через $u_i(j)$ элемент с номером j из U_i, $j = 1, \ldots, |U_i|$, т. е. $\bigcup\limits_{j=1}^{|U_i|} \{u_i(j)\} = U_i$.

Решение задачи $P|\mathrm{set}_i,\ \Delta\mathrm{s}_{luq}|C_{\max}$ будем представлять генотипом ζ, состоящим из двух подстрок: подстроки назначений ξ и подстроки по-

рядка η, а именно, $\zeta = (\xi, \eta) \in \Omega$, где $\Omega = B \times C,\ B = \{0,1\}^d,\ C = \mathbb{Z}_+^d$. Здесь и далее $\mathbb{Z}_+$ обозначает множество неотрицательных целых чисел.

Подстрока назначений $\xi = (\xi_{11}, \ldots, \xi_{1|U_1|}, \ldots, \xi_{k1}, \ldots, \xi_{k|U_k|}) \in \{0,1\}^d$ определяет технологии, участвующие в производстве продуктов:

$$\xi_{ij} = \begin{cases} 1, & \text{если технология } u_i(j) \text{ производства продукта } i \text{ используется,} \\ 0 & \text{в противном случае.} \end{cases}$$

Подстрока порядка $\eta = (\eta_{11}, \ldots, \eta_{1|U_1|}, \ldots, \eta_{k1}, \ldots, \eta_{k|U_k|}) \in \mathbb{Z}_+^d$ определяет порядок выполнения технологий:

$$\eta_{ij} = \begin{cases} \text{номеру точки событий для технологии } u_i(j), \text{ если } \xi_{ij} = 1, \\ 0 \text{ в противном случае.} \end{cases}$$

Назначение генов ξ_{ij} и η_{ij}, где $i = 1, \ldots, k,\ j = 1, \ldots, |U_i|$, однозначно определяет значения булевых переменных (w_{un}) в модели (2)–(12), а именно,

$$w_{u_i(j),n} = \begin{cases} 1, \text{ если } \eta_{ij} = n, \\ 0, \text{ если } \eta_{ij} = 0, \end{cases} \qquad (13)$$
$$n = 1, \ldots, n_{\max},\ i = 1, \ldots, k,\ j = 1, \ldots, |U_i|.$$

Значения вещественных переменных и целевой функции $C_{\max}$ можно найти с помощью решения задачи линейного программирования $\mathcal{P}_{ЛП}$. Таким образом, схема представления решений полностью определена.

Построение начальной популяции. При построении особи начальной популяции сначала строится подстрока назначений ξ, по которой с помощью специальной процедуры строится подстрока порядка η. Построение ξ происходит по следующей схеме: для каждого продукта независимо от остальных случайным образом с равномерным распределением выбирается одна из технологий, пригодных для его производства. Был опробован также вариант, в котором число технологий для производства каждого продукта $i \in I$ выбирается с равномерным распределением от 1 до $|U_i|$. Однако экспериментальное сравнение показало, что ГА показывает лучшие результаты при выборе одной технологии для каждого продукта при построении особей популяции Π^0.

Для того чтобы по подстроке назначений ξ сформировать подстроку порядка η, каждой технологии $u_i(j)$, для которой $\xi_{ij} = 1,\ i = 1, \ldots, k,\ j = 1, \ldots, |U_i|$, поставим в соответствие отдельную точку событий. Далее определим порядок следования технологий. Для этого построим вспомогательный полный ориентированный граф $G = (U', A)$ с множествами

вершин $U' = \{1, \ldots, d'\}$ и дуг A, где $d' = \sum\limits_{i=1}^{k} \sum\limits_{j=1}^{|U_i|} \xi_{ij}$. Вершины графа соответствуют выбранным технологиям, а длина дуги равна максимальной длительности переналадки машин между соответствующими технологиями. Далее для этого графа решаем задачу о кратчайшем гамильтоновом пути, например, с помощью эвристики «присоединение к пути». Для описания данной эвристики введём следующие обозначения:

$\alpha_{jj'}$ — длина дуги из j в j' при $j \neq j'$, $\alpha_{jj} = +\infty$, $j = 1, \ldots, d'$;

$\pi = (\pi_1, \ldots, \pi_{d'})$ — перестановка, определяющая порядок следования вершин, а именно, π_j — номер j-й вершины в гамильтоновом пути в графе G;

$\pi^\tau = (\pi_1, \ldots, \pi_\tau)$ — перестановка, полученная на шаге $\tau = 1, \ldots, d'$;

$S(\pi)$ — суммарная длина пути, определяемого перестановкой π.

Алгоритм «присоединение к пути»

Шаг 1. Положить $\pi_1 := j$, где $j \in \{1, \ldots, d'\}$ такое, что $\min\limits_{j' \neq j} \alpha_{j'j}$ максимален. Положить $U' := U' \setminus \{\pi_1\}$ и перейти на шаг 2.

Шаг τ ($\tau = 2, \ldots, d'$). Для каждой вершины $v \in U'$ вычислить значение $S_v = \min\limits_{j=2,\ldots,\tau} S(\pi_1, \ldots, \pi_{j-1}, v, \pi_{j+1}, \ldots, \pi_{\tau-1})$. Позицию j, на которой достигается минимум S_v, обозначить через j_v. Вершину $v \in U'$, имеющую минимальное значение S_v, вставить в позицию j_v в перестановку $\pi^{\tau-1}$ со сдвигом вправо содержимого позиций $j_v, \ldots, \tau - 1$ и положить $U' := U' \setminus \{v\}$. Если $\tau < d'$, то перейти на шаг $\tau + 1$, иначе $\pi^* := \pi^{d'}$.

Для задачи $P|\mathrm{set}_i, \mathrm{pmtn}, \Delta\mathrm{s}_{luq}|C_{\max}$ может не существовать оптимального решения, в котором каждая технология используется не более одного раза, так как дробное хроматическое число может не совпадать с хроматическим числом графа (см. пример в [23]). Поэтому не гарантируется существование генотипа, представляющего оптимальное решение этой задачи. Однако для задачи $P|\mathrm{set}_i,\ \Delta\mathrm{s}_{luq}|C_{\max}$, представленной моделью (2)–(12), в любом оптимальном расписании выполнение какой-либо технологии может быть предписано не более чем в одной точке событий, и, кроме того, имеет место

Утверждение 5. *Для задачи* (2)–(12) *найдётся оптимальное решение, в котором различным технологиям соответствуют различные точки событий.*

Таким образом, всегда существует генотип, представляющий оптимальное решение задачи $P|\mathrm{set}_i,\ \Delta\mathrm{s}_{luq}|C_{\max}$.

Отметим, что при построении подстроки порядка η по подстроке назначений ξ можно предусмотреть процедуру, объединяющую в одной точке событий несколько технологий. Опишем один из возможных способов такого объединения. Построим вспомогательный неориентированный граф $G' = (V', E)$, где V' — множество вершин, E — множество рёбер. Вершины данного графа соответствуют тем технологиям $u_i(j)$, для которых $\xi_{ij} = 1,\ i = 1, \ldots, k,\ j = 1, \ldots, |U_i|$, а наличие ребра между вершинами указывает на то, что соответствующие технологии не могут выполняться в одной точке событий, поскольку используют хотя бы одну общую машину. Далее для построенного графа решаем задачу о минимальной вершинной раскраске с помощью градиентного алгоритма [4]. Допустимое решение представленной задачи определяет для каждой технологии точку событий, в которой она будет выполняться: технологии, соответствующие вершинам, раскрашенным в один цвет, выполняются в одной точке событий.

Порядок следования технологий может быть определён с помощью эвристики «присоединение к пути» в графе, где вершины соответствуют множествам технологий с общей точкой событий, а длина дуги равна максимальной длительности переналадки машин между соответствующими технологиями.

Данная модификация не гарантирует существования генотипа, представляющего оптимальное решение задачи $P|\mathrm{set}_i,\ \Delta \mathrm{s}_{luq}|C_{\max}$, но позволяет сократить количество используемых точек событий, а следовательно, уменьшить размерность подзадачи линейного программирования $\mathcal{P}_{ЛП}$.

Оператор селекции. В качестве оператора селекции будем использовать s-турнирную селекцию: из популяции Π^t независимо извлекается s особей с равномерным распределением и из них выбирается лучшая по целевой функции.

Оператор мутации. Оператор мутации вносит в генотип некоторые случайные изменения [7]. Данный оператор применяется как к подстроке назначений ξ, так и к подстроке порядка η. Мутация для подстроки η: с вероятностью P_η случайным образом выбираем две точки событий и меняем их содержимое местами, иначе (с вероятностью $1 - P_\eta$) подстрока η остаётся без изменений.

Мутация для подстроки ξ происходит следующим образом. Для каждого продукта $i \in I$ независимо от всех остальных с вероятностью P_ξ выполняем:

(a) если $\sum_{j=1}^{|U_i|} \xi_{ij} > 1$, то выбираем с равномерным распределением один из генов $\xi_{i1}, \xi_{i2}, \ldots, \xi_{i|U_i|}$ и инвертируем его значение;

(b) если $\sum_{j=1}^{|U_i|} \xi_{ij} = 1$, то выбираем с равномерным распределением один из генов $\xi_{i1}, \xi_{i2}, \ldots, \xi_{i|U_i|}$, имеющих значение 0, и заменяем его на 1.

Иначе (с вероятностью $1 - P_\xi$) множество генов ξ_{ij}, $j \in \{1, \ldots, |U_i|\}$, остаётся без изменений.

Если в результате мутации подстрока назначений изменяется, то соответствующая ей подстрока порядка строится так же, как и при построении особей начальной популяции.

Оператор кроссинговера. Под действием оператора кроссинговера две родительские особи обмениваются своими признаками, порождая одного или двух потомков [7]. В настоящей статье предлагается два варианта оператора кроссинговера: равномерный кроссинговер и оптимальная рекомбинация (см. разд. 3).

Оператором равномерного кроссинговера строится пара потомков. При этом сначала формируется подстрока назначений для каждого из них, а соответствующая подстрока порядка определяется так же, как и при построении особей начальной популяции. С заданной вероятностью P_c равномерный кроссинговер выполняет следующее преобразование подстрок ξ^1 и ξ^2. Для каждого продукта независимо от остальных используем набор технологий его производства от одного из родителей, при этом выбор родителей равновероятен. Иначе (с вероятностью $1 - P_c$) подстроки назначений в генотипах потомков полагаются равными ξ^1 и ξ^2.

Очевидно, что операторы построения начальной популяции, кроссинговера и мутации генерируют только генотипы, представляющие допустимые решения задачи $P|\mathrm{set}_i,\ \Delta\mathrm{s}_{luq}|C_{\max}$.

Заметим, что предложенный алгоритм может быть применен к задаче $P|\mathrm{set}_i,\ \mathrm{s}_{luq}|C_{\max}$ без предположения о выполнении неравенства треугольника (1). При этом вместо модели (2)–(12) используется модель из [6] совместно с неравенством (12).

Сходимость алгоритма. Исследуем сходимость ГА с равномерным кроссинговером к оптимуму. Для этого воспользуемся теоремой о сходимости эволюционных алгоритмов (ЭА). Приведём описание общего вида операторов ЭА и схемы ЭА.

Рассмотрим задачу минимизации: найти $f^* = \min\{f(x) :\ x \in X\}$, где

X — конечное пространство решений. Обозначим популяцию поколения t, $t = 0, 1, 2, \ldots$, в ЭА через $\Pi^t = (x^{1,t}, x^{2,t}, \ldots, x^{r,t}) \in X^r$, и пусть $\widehat{\Pi} = \bigcup_{j=1}^{\widetilde{r}} \{x^j\}$ для любой популяции $\Pi = (x^1, \ldots, x^{\widetilde{r}})$. Каждая новая популяция Π^{t+1} строится на основании текущей Π^t с помощью следующих вероятностных операторов. Здесь под вероятностным оператором будем понимать рандомизированную процедуру [5], результат которой представляет собой случайную величину, зависящую только от аргумента процедуры и исходных данных задачи.

Оператором *отбора* $\delta : X^r \to X^{r'}$ извлекаются r' копий родительских решений из текущей популяции. Если $\Pi' = \delta(\Pi^t)$, то $\widehat{\Pi}' \subseteq \widehat{\Pi}^t$.

Оператор *воспроизведения* $\rho : X^{r'} \to X^{r''}$ вносит некоторые случайные изменения в родительские решения, создавая при этом r'' решений-потомков, составляющих популяцию Π''.

Оператором *выживания* $\sigma : X^r \times X^{r''} \to X^r$ определяются решения из Π^t и их потомки из Π'', которые составят популяцию Π^{t+1}, т. е. $\widehat{\sigma}(\Pi^t, \Pi'') \subseteq \widehat{\Pi}^t \cup \widehat{\Pi}''$.

Работа ЭА начинается со случайной популяции Π^0 и продолжается итерациями композиции $\Pi^{t+1} = \sigma(\rho(\delta(\Pi^t)), \Pi^t)$.

Будем говорить, что ЭА *сходится к оптимуму почти наверное* при $t \to \infty$, если $\min\limits_{j=1,\ldots,r} f(x^{j,t}) \stackrel{t\to\infty}{\longrightarrow} f^*$ с вероятностью 1. Следующая теорема задаёт достаточные условия сходимости ЭА к оптимуму почти наверное (доказывается аналогично теореме 2.1 в [28]).

Теорема 1. *Пусть операторы* ЭА *обладают следующими свойствами: существует величина* $\varepsilon > 0$ *такая, что*

($\mathcal{B}_1$) $P\{x \in \widehat{\delta}(\Pi)\} > \varepsilon$ *при любых* $\Pi \in X^r$, $x \in \widehat{\Pi}$;

($\mathcal{B}_2$) $P\{x \in \widehat{\sigma}(\Pi, \Pi'')\} > \varepsilon$ *при любых* $\Pi \in X^r$, $\Pi'' \in X^{r''}$, $x \in \widehat{\Pi}''$;

($\mathcal{B}_3$) *для любого* $x \in X$ *найдётся последовательность решений* y_1, $y_2, \ldots, y_\lambda$ *такая, что* $y_1 = x$, $f(y_\lambda) = f^*$ *и при всех* $j = 1, \ldots, \lambda - 1$ *если* $y_j \in \widehat{\Pi}'$, *то* $P\{y_{j+1} \in \widehat{\rho}(\Pi')\} > \varepsilon$;

($\mathcal{B}_4$) $\min\{f(x) : x \in \widehat{\sigma}(\Pi, \Pi'')\} = \min\{f(x) : x \in \widehat{\Pi} \cup \widehat{\Pi}''\}$ *при любых* $\Pi \in X^r$, $\Pi'' \in X^{r''}$.

Тогда ЭА сходится к оптимуму почти наверное при $t \to \infty$.

Теорема 2. *Если* $0 < P_\xi < 1$, $0 < P_\eta \leqslant 1$, $0 \leqslant P_c < 1$, *то при решении задачи* $P|\mathrm{set}_i, \Delta\mathrm{s}_{luq}|C_{\max}$ *ГА с равномерным кроссинговером сходится к оптимуму почти наверное при* $t \to \infty$.

Доказательство. Определим вид операторов ЭА в исследуемом ГА и свойства, которыми они обладают.

(i) При действии оператора отбора $\delta(\Pi^t)$ выбираются родительские особи $p^1,\ p^2$ с помощью s-турнирной селекции. Ясно, что для данного оператора выполнено свойство $(\mathcal{B}_1)$.

(ii) Оператор воспроизведения $\rho(\Pi')=(c^1, c^2)$ состоит в последовательном применении мутации и равномерного кроссинговера к родительским особям $p^1,\ p^2$. Докажем, что для данного оператора выполнено свойство $(\mathcal{B}_3)$. Ввиду утверждения 5 оптимальное решение всегда представимо с помощью некоторого генотипа $\zeta^* = (\xi^*, \eta^*)$. Покажем, что для любого генотипа $\zeta = (\xi, \eta)$ существует положительная вероятность перехода из ζ в ζ^* за конечное число шагов. Так как $0 < P_\xi < 1$, $0 \leqslant P_c < 1$, перейти от генотипа $\zeta = (\xi, \eta)$ к $\widetilde{\zeta} = (\widetilde{\xi}, \widetilde{\eta})$, где $\widetilde{\xi} = \xi^*$, можно за конечное число шагов с положительной вероятностью при мутациях только в подстроке назначений. Далее, так как $0 < P_\eta \leqslant 1$, перейти от $\widetilde{\zeta}$ к ζ^* можно за конечное число шагов с положительной вероятностью при мутациях только в подстроке порядка $(P_\xi < 1,\ P_c < 1)$.

(iii) Оператор выживания $\sigma(\Pi^t, \Pi'')$ выбирает с равномерным распределением два генотипа среди имеющих значение целевой функции выше среднего в популяции Π^t и заменяет их построенными оператором $\rho(\Pi')$ потомками $c^1,\ c^2$. Легко проверить, что для данного оператора выполнены свойства $(\mathcal{B}_2)$ и $(\mathcal{B}_4)$.

Таким образом, исследуемый ГА сходится к оптимуму почти наверное по теореме 1. Теорема 2 доказана.

3. Генетический алгоритм с оптимальной рекомбинацией

Для решения задачи $P|\mathrm{set}_i,\ \Delta\mathrm{s}_{lrq}|C_{\max}$ может быть использован генетический алгоритм с *оптимальной рекомбинацией* [12], основанный на решении серии подзадач вида (2)–(12), в которых часть булевых переменных исходной задачи фиксирована.

Используя схему представления решений из разд. 2, сформулируем *задачу оптимальной рекомбинации* для $P|\mathrm{set}_i,\ \Delta\mathrm{s}_{lrq}|C_{\max}$: для произвольных заданных родительских генотипов $p^1 = (\xi^1, \eta^1)$, $p^2 = (\xi^2, \eta^2)$, представляющих допустимые решения, требуется построить представляющий допустимое решение генотип $\zeta' = (\xi', \eta')$ такой, что

(i) $\xi'_{ij} = \xi^1_{ij}$ или $\xi'_{ij} = \xi^2_{ij}$ для всех $i = 1, \ldots, k$, $j = 1, \ldots, |U_i|$;

(ii) $\eta'_{ij} = \eta^1_{ij}$ или $\eta'_{ij} = \eta^2_{ij}$ для всех $i = 1, \ldots, k$, $j = 1, \ldots, |U_i|$;

(iii) ζ' имеет минимальное значение целевой функции среди всех генотипов, удовлетворяющих условиям (i) и (ii).

В терминах модели (2)–(12) условия (i) и (ii) соответствуют фиксации тех булевых переменных, которые совпадают при отображении родительских генотипов в пространство решений (2)–(12) согласно (13). Обозначим через $\mathcal{P}_{ЧЦЛП}$ задачу оптимальной рекомбинации, записанную в виде (2)–(12) с фиксацией булевых переменных, совпадающих в родительских решениях. Для решения задачи $\mathcal{P}_{ЧЦЛП}$ применяется метод ветвей и границ или некоторая его модификация.

При высокой вероятности мутации возникают вспомогательные подзадачи $\mathcal{P}_{\text{ЧЦЛП}}$ большой размерности, тогда как при низкой вероятности мутации подзадачи $\mathcal{P}_{ЧЦЛП}$ часто оказываются тривиальными. Для исключения тривиальных ситуаций и поддержания размерностей вспомогательных подзадач в границах, где методом ветвей и границ оптимальное решение может быть получено за достаточно короткое время, нами использован следующий механизм адаптации [12]. Начальные значения вероятностей мутации полагаются заданными $P_\xi^0 > 0,\ P_\eta^0 > 0$, а затем адаптируются в процессе работы алгоритма. Если за отведённое время T_{rec} методом ветвей и границ удаётся найти оптимальное решение подзадачи $\mathcal{P}_{ЧЦЛП}$, то $P_\xi := 1.1 \cdot P_\xi,\ P_\eta := 1.1 \cdot P_\eta$. Если же оптимальное решение не найдено и $P_\xi > P_\xi^0,\ P_\eta > P_\eta^0$, то $P_\xi := P_\xi/2,\ P_\eta := P_\eta/2$.

Если за время T_{rec} удаётся найти допустимое решение вспомогательной подзадачи $\mathcal{P}_{ЧЦЛП}$, но полученный генотип совпадает с одним из уже имеющихся в текущей популяции или допустимое решение не найдено, то в качестве результата оператора оптимальной рекомбинации используется генотип, полученный применением мутации к первой родительской особи.

Поскольку результатом оптимальной рекомбинации является один генотип, а не два, инструкции шагов 2.3 и 2.4 в ГА модифицируются соответствующим образом.

Теорема 3. *Если* $0 < P_\xi < 1,\ 0 < P_\eta \leqslant 1$ *и* T_{rec} *достаточно велико, то при решении задачи* $P|\text{set}_i, \Delta\text{s}_{luq}|C_{\max}$ ГА *с оптимальной рекомбинацией сходится к оптимуму почти наверное при* $t \to \infty$.

Доказательство. Воспользуемся теоремой 1. Свойства $(\mathcal{B}_1)$, $(\mathcal{B}_2)$ и $(\mathcal{B}_4)$ доказываются как в теореме 2.

Оператор воспроизведения $\rho(\Pi')=c^1$ состоит в последовательном применении мутации и оптимальной рекомбинации к родительским особям p^1 и p^2. Докажем свойство $(\mathcal{B}_3)$.

Поскольку родительские особи $p^1,\ p^2$ выбираются независимо друг от друга, причём каждая особь из текущей популяции может быть выбрана в качестве родительской, то вероятность того, что произвольная

особь из текущей популяции будет выбрана одновременно в качестве p^1 и p^2, положительна. Вероятность того, что после независимого применения операторов мутации к p^1 и $p^2 = p^1$ будет получена одна и та же особь $\widetilde{c}^1$, также положительна. Тогда результатом оптимальной рекомбинации будет генотип $c^1 = \widetilde{c}^1$. Дальнейшие рассуждения аналогичны теореме 2. Теорема 3 доказана.

4. Вычислительный эксперимент

Для оценки возможностей предложенных генетических алгоритмов и сравнения их с коммерческим пакетом решения задач частично целочисленного линейного программирования CPLEX 12.1 проведён вычислительный эксперимент на случайным образом сгенерированных задачах четырёх серий $S1$, $S2$, $S3$ и $S4$. Сравнение проводилось по качеству приближённого решения, найденного за отведённый промежуток времени.

Модели частично целочисленного линейного программирования для задач $P|\mathrm{set}_i,\ \Delta\mathrm{s}_{luq}|C_{\max}$ и $P|\mathrm{set}_i,\ \mathrm{pmtn},\ \Delta\mathrm{s}_{luq}|C_{\max}$ записаны в системе моделирования GAMS 23.2, и для решения этих задач применялся пакет CPLEX. При этом использовался метод ветвей и границ с отсечениями, настройки которого были выбраны по умолчанию кроме параметра mipemphasis=4, устанавливающего приоритет на быстрый поиск приближённых решений. Генетические алгоритмы также реализованы в GAMS 23.2. Тестирование проводилось на ЭВМ Intel Core 2 Duo CPU E7200 2.54 ГГц, оперативная память 2 Гб.

При генерации тестовых задач для каждой технологии $u \in U$ число машин $|M_u| \in \{1, \ldots, m\}$ выбиралось случайно с равномерным распределением, а затем машины случайно назначались на данную технологию без повторений. Числовые значения для всех тестовых задач генерировались случайным образом с равномерным распределением из следующих множеств: $|U_i| \in \{1, \ldots, U_{\max}\}$; $V_i \in [1, V_{\max}]$; $a_u \in \left[1, \frac{V_i}{2}\right]$, где i такое, что $u \in U_i$; $s_{luq} \in [0, s_{\max}]$. В табл. 1 приведены выбранные значения параметров для каждой серии.

Для иллюстрации сложности полученных задач в табл. 2 представлены минимальная и максимальная размерности задач в каждой серии, записанных в модели (2)–(11). В модели (2)–(12) число ограничений увеличивается на величину d.

Опишем настройки ГА с равномерным кроссинговером (GA) и ГА с оптимальной рекомбинацией (GA_{or}).

Алгоритмы GA и GA_{or} запускались многократно в течение заданного промежутка времени согласно следующей схеме: пусть θ — номер

итерации, на которой получено улучшение рекордного значения целевой функции. GA (GA_{or}) перезапускается, если за последующие θ итераций не получено улучшения рекорда и число итераций превышает $0.8r$.

Таблица 1

Параметры серий

серия	число задач	k	m	$U_{\max}$	$V_{\max}$	$s_{\max}$	$n_{\max}$
$S1$	30	8	10	5	50	10	6
$S2$	30	9	10	6	30	15	7
$S3$	30	12	10	8	25	20	7
$S4$	30	15	12	8	25	40	8

Таблица 2

Минимальная и максимальная размерности задач в сериях

	ограничений		переменных			
			вещественных		булевых	
серия	min	max	min	max	min	max
$S1$	6980	10230	277	433	138	216
$S2$	13269	19333	491	715	245	357
$S3$	36572	47850	925	1289	462	644
$S4$	84323	108060	1279	1841	648	920

Значения вещественных переменных и целевой функции находятся в алгоритме GA решением задачи линейного программирования $\mathcal{P}_{ЛП}$ с помощью пакета CPLEX. Задача частично целочисленного линейного программирования $\mathcal{P}_{ЧЦЛП}$, возникающая при оптимальной рекомбинации в алгоритме GA_{or}, также решается с помощью пакета CPLEX (с параметром mipemphasis=1). При этом устанавливается максимальное время $T_{\text{rec}} = 3$ сек. Размер s турнира равен 5, размер r популяции — 50.

Поведение ГА существенно зависит от значений его параметров. Для определения значений вероятностей мутации P_ξ, P_η и вероятности кроссинговера P_c в алгоритме GA проведён предварительный вычислительный эксперимент. В табл. 3 представлены выбранные величины. Начальные значения вероятностей мутации в алгоритме GA_{or} полагаются равными $P_\xi^0 = 0.001,\ P_\eta^0 = 0.001$.

Для проведения вычислительного эксперимента установлено время работы алгоритмов, одно и то же для всех примеров: для серии $S1$ — 900 с, $S2$ — 1500 с, $S3$ — 3600 с и $S4$ — 5400 с.

При описании результатов вычислительного эксперимента будут использоваться следующие обозначения:

$\Delta_{\max}$, Δ_{avr} — максимальная и средняя относительные погрешности при сравнении с наилучшим известным из экспериментов значением целевой функции соответственно;

NF, NB — количество задач, для которых получено допустимое решение и решение с наилучшим известным значением целевой функции соответственно.

Т а б л и ц а 3

Значения вероятностей мутации и кроссинговера в GA

серия	P_ξ	P_η	P_c
$S1$	0.15	0.15	0.75
$S2$	0.15	0.15	0.85
$S3$	0.05	0.1	0.85
$S4$	0.05	0.15	0.75

Отметим, что при решении задач всех серий пакетом CPLEX за установленное время не удаётся доказать оптимальность полученного решения. Алгоритм GA (GA_{or}), в котором предусмотрена процедура, объединяющая технологии по точкам событий, описанная в разд. 2, обозначим через GA^{gt} (GA^{gt}_{or}).

Т а б л и ц а 4

Результаты работы алгоритмов для серий $S1$, $S2$

алгоритм	серия $S1$				серия $S2$			
	$\Delta_{\max}$	Δ_{avr}	NF	NB	$\Delta_{\max}$	Δ_{avr}	NF	NB
CPLEX (pmtn)	0.037	0.002	30	26	0.206	0.011	30	25
CPLEX (npmtn)	0.037	0.002	30	26	0.206	0.012	30	25
GA^{gt}	0.368	0.054	30	24	0.253	0.032	30	23
GA^{gt}_{or}	0	0	30	30	0	0	30	30
GA	0.548	0.064	30	22	0.353	0.077	30	20
GA_{or}	0.016	0.001	30	27	0.035	0.002	30	26

В табл. 4 и 5 представлены результаты работы алгоритмов. Здесь строка CPLEX (pmtn) соответствует результатам пакета CPLEX при решении задач по модели (2)–(11), когда допускается прерывание выполнения технологий, а строка CPLEX (npmtn) — результатам пакета при решении задач по модели (2)–(12), когда прерывание выполнения технологий не допускается. Рассмотрим детально результаты, полученные для задачи $P|\text{set}_i,\ \Delta\text{s}_{luq}|C_{\max}$.

Для выявления статистически значимых различий между выборками, состоящими из значений целевой функции, полученных исследуемыми алгоритмами, использовался тест Вилкоксона [2]. Далее под статистически значимыми различиями будем подразумевать значимые различия между выборками по данному тесту при уровне значимости 0.05.

Таблица 5

Результаты работы алгоритмов для серий $S3$, $S4$

алгоритм	серия $S3$				серия $S4$			
	$\Delta_{\max}$	Δ_{avr}	NF	NB	$\Delta_{\max}$	Δ_{avr}	NF	NB
CPLEX (pmtn)	0.104	0.014	30	22	0.283	0.029	29	22
CPLEX (npmtn)	0.104	0.016	30	22	0.294	0.032	29	22
GA^{gt}	0.149	0.026	30	20	0.205	0.036	30	20
GA^{gt}_{or}	0	0	30	30	0	0	30	30
GA	0.329	0.061	30	16	0.331	0.082	30	17
GA_{or}	0.071	0.005	30	26	0.073	0.01	30	24

Алгоритм GA^{gt}_{or} на всех сериях получает наилучшее известное нам значение целевой функции во всех задачах. Его преимущество над GA^{gt} и GA подтверждается статистически значимыми различиями на всех сериях, а над CPLEX — на всех сериях кроме $S1$. В то же время преимущество GA^{gt}_{or} над GA_{or} подтверждается статистически значимыми различиями только на серии $S4$.

Алгоритм GA_{or}, как видно из табл. 4 и 5, показывает лучшие результаты, чем CPLEX, а также генетические алгоритмы GA^{gt} и GA, из чего можно сделать вывод о перспективности ГА с оптимальной рекомбинацией не только при использовании процедуры объединения технологий по точкам событий, но и без неё.

Результаты пакета CPLEX на всех сериях несколько лучше, чем у алгоритмов GA^{gt} и GA. Преимущество CPLEX над GA подтверждается статистически значимыми различиями на всех сериях кроме $S4$, а над GA^{gt} — только на серии $S2$.

На рис. 1 представлена частота получения наилучшего известного из экспериментов значения целевой функции для каждого из алгоритмов. Из рис. 1 и табл. 4, 5 видно, что GA^{gt}_{or} показывает устойчивое преимущество над остальными алгоритмами, которое становится значительнее с ростом размерности задач. Алгоритм GA_{or} даёт лучшие результаты, чем CPLEX, а алгоритмы GA^{gt} и GA несколько уступают пакету CPLEX. Также обратим внимание на то, что применение процедуры объединения

технологий по точкам событий увеличило частоту получения наилучшего известного решения.

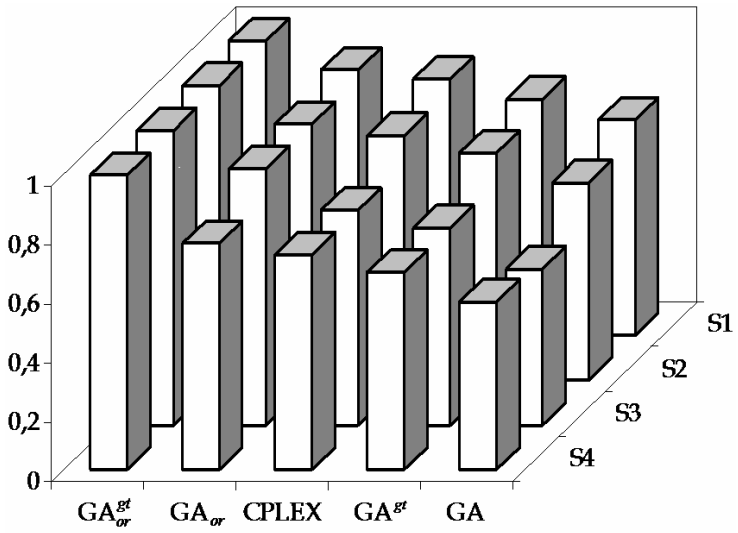

Рис. 1. Частота получения наилучшего известного решения

Как видно из табл. 4 и 5, аналогичные результаты имеют место для задачи $P|\text{set}_i,\ \text{pmtn},\ \Delta\text{s}_{luq}|C_{\max}$. Выявленные ранее статистически значимые различия сохраняются. Таким образом, при решении задач по модели (2)–(11) пакету CPLEX за установленное время не удаётся найти решение лучшего качества за счёт возможности прерывания выполнения технологий. Это может свидетельствовать о том, что для получения решений лучшего качества необходимо больше времени или выбранного числа точек событий недостаточно.

В разд. 1 получена оценка d^m количества точек событий, заведомо достаточного для получения оптимального решения задачи $P|\text{set}_i, \text{pmtn}, \Delta\text{s}_{luq}|C_{\max}$. Однако использование d^m точек событий в модели (2)–(11) оказывается неэффективным при решении практически значимых задач ввиду их большой размерности в данной модели. Так, например, увеличение числа точек событий в два раза приводит к следующему. На сериях $S1,\ S2$ в двух тестовых примерах CPLEX за установленное время удаётся найти решение с меньшим значением целевой функции, чем у решений, полученных алгоритмом GA_{or}^{gt}, однако на большинстве задач этих серий результаты пакета ухудшаются. На всех задачах серий $S3,\ S4$

при увеличении числа точек событий в два раза CPLEX получает решения худшего качества, чем ранее. Увеличение числа точек событий до величины d на всех сериях приводит к дальнейшему ухудшению результатов пакета CPLEX при том же ограничении на время вычислений.

Отметим, что на всех сериях нижние оценки оптимума, полученные пакетом CPLEX за установленное время, в большинстве случаев оказались равными нулю. В связи с большим разрывом между значениями целевой функции полученных решений и нижними оценками, выяснение оптимальных значений целевой функции в задачах большой размерности представляется чрезмерно трудоёмким. Однако для серии $S1$ оптимальные решения найдены в дополнительном вычислительном эксперименте методом ветвей и границ из пакета CPLEX без ограничения времени счёта. При этом использовались модели (2)–(11) и (2)–(12) при $n_{\max} = 6$. Время счёта составило от 4 до 23 часов при работе пакета CPLEX в режиме с двумя процессорами.

Результаты данного эксперимента показали, что алгоритм GA_{or}^{gt} получает оптимальное решение задачи (2)–(12) при $n_{\max} = 6$ во всех 30 тестовых примерах, алгоритм GA_{or} — в 27, GA^{gt} — в 24 и GA — в 22. Таким образом, в тех случаях, когда исследуемым алгоритмам удавалось найти наилучшее значение целевой функции, это значение являлось оптимальным в модели (2)–(12) при $n_{\max} = 6$. Кроме того, в 29 тестовых примерах из 30 при $n_{\max} = 6$ оптимальные значения целевой функции в моделях (2)–(11) и (2)–(12) равны между собой, а в одном примере отличаются примерно на 1.7%. В итоге алгоритм GA_{or}^{gt} получает оптимальное решение задачи (2)–(11) при $n_{\max} = 6$ в 29 тестовых примерах, алгоритм GA_{or} — в 26, GA^{gt} — в 23 и GA — в 21.

Тем самым можно сделать вывод о целесообразности использования GA_{or}^{gt} и GA_{or} для нахождения за короткое время приближённого решения задач $P|\mathrm{set}_i,\ \Delta\mathrm{s}_{luq}|C_{\max}$ и $P|\mathrm{set}_i,\ \mathrm{pmtn},\ \Delta\mathrm{s}_{luq}|C_{\max}$ по крайней мере в случаях, когда исходные данные аналогичны рассмотренным в вычислительном эксперименте.

5. Заключение

В статье рассмотрена задача составления расписаний с группировкой машин по технологиям. Особенностью постановки является то, что каждый продукт имеет несколько технологий производства, при выполнении которых используется сразу несколько машин, работающих одновременно. Построена модель частично целочисленного линейного программирования для данной задачи. Установлены новые полиномиально разрешимые и NP-трудные в сильном смысле частные случаи.

Для приближённого решения задачи разработаны генетический алгоритм с равномерным кроссинговером и генетический алгоритм с оптимальной рекомбинацией, основанные на решении серии подзадач пониженной размерности в соответствии с предложенной моделью. Исследована сходимость данных алгоритмов. Проведён вычислительный эксперимент на построенных случайным образом тестовых примерах, показавший существенное преимущество генетического алгоритма с оптимальной рекомбинацией по сравнению с непосредственным применением пакета CPLEX и с генетическим алгоритмом, использующим равномерный кроссинговер.

Авторы благодарят рецензента и П. А. Борисовского за ценные замечания.

ЛИТЕРАТУРА

1. **Борисовский П. А.** Генетический алгоритм для одной задачи составления производственного расписания с переналадками // Тр. XIV Байкальской междунар. школы-семинара «Методы оптимизации и их приложения». — Иркутск: ИСЭМ СО РАН, 2008. — Т. 4. — С. 166–173.
2. **Гмурман В. Е.** Теория вероятностей и математическая статистика: Уч. пос. — М.: Высшее образование, 2006. — 479 с.
3. **Гэри М., Джонсон Д.** Вычислительные машины и труднорешаемые задачи. — М.: Мир, 1982. — 416 с.
4. **Ильев В. П.** Оценки погрешности приближённого алгоритма для задачи о раскраске графа // Тр. XIII Байкальской междунар. школы-семинара «Методы оптимизации и их приложения». — Иркутск: ИСЭМ СО РАН, 2005. — Т. 1. — С. 491–495.
5. **Китаев А., Шень А., Вялый М.** Классические и квантовые вычисления. — М.: МЦНМО, ЧеРо, 1999. – 192 с.
6. **Коваленко Ю. В.** Модель с непрерывным представлением времени для задачи составления расписаний многопродуктового производства // Препринт arxiv: 1105.2437 [math.OC]. — Cornell: Cornell University, 2011. — 8 с.
7. **Рутковская Д., Пилиньский М., Рутковский Л.** Нейронные сети, генетические алгоритмы и нечёткие системы. — М.: Горячая линия–Телеком, 2006. — 452 с.
8. **Танаев В. С., Ковалев М. Я., Шафранский Я. М.** Теория расписаний. Групповые технологии. — Минск: Ин-т техн. кибернетики НАН Беларуси, 1998. — 290 с.
9. **Хачиян Л. Г.** Полиномиальные алгоритмы в линейном программировании // Докл. АН СССР. — 1979. — Т. 244. — С. 1093–1096.
10. **Bianco L., Blazewicz J., Dell'Ohno P., Drozdowski M.** Scheduling multiprocessor tasks on a dynamic configuration of dedicated processors // Ann. Oper. Res. — 1995. — Vol. 58. — P. 493–517.

11. **Blazewicz J., Dell'Ohno P., Drozdowski M., Speranza M. G.** Scheduling multiprocessor tasks on three dedicated processors // Inf. Process. Lett. — 1992. — Vol. 41. — P. 275–280; Corrigendum: Vol. 49. — P. 269–270.
12. **Borisovsky P. A., Dolgui A., Eremeev A. V.** Genetic algorithms for a supply management problem: MIP–recombination vs greedy decoder // Eur. J. Oper. Res. — 2009. — Vol. 195, N 3. — P. 770–779.
13. **Dolgui A., Eremeev A. V., Kovalyov M. Y.** Multi-product lot-sizing and scheduling on unrelated parallel machines // Res. Report No. 2007–500–011. — St.-Etienne: Ecole des Mines de St.-Etienne, 2007. — 15 p.
14. **Drozdowski M.** Scheduling multiprocessor tasks — an overview // Eur. J. Oper. Res. — 1996. — Vol. 94. — P. 215–230.
15. **Feige U., Kilian J.** Zero knowledge and the chromatic number // J. Comput. Syst. Sci. — 1998. — Vol. 57, N 2. — P. 187–199.
16. **Floudas C. A., Kallrath J., Pitz H. J., Shaik M. A.** Production scheduling of a large-scale industrial continuous plant: short-term and medium-term scheduling // Comp. Chem. Eng. — 2009. — Vol. 33. — P. 670–686.
17. **Grötschel M., Lovasz L., Schrijver A.** The ellipsoid method and its consequences in combinatorial optimization // Combinatorica. — 1981. — Vol. 1. — P. 169–197.
18. **Håstad J.** Clique is hard to approximate within $n^{1-\varepsilon}$ // Acta Math. — 1999. — Vol. 182. — P. 105–142.
19. **Holland J. H.** Adaptation in natural and artificial systems. — Ann Arbor: Univ. Michigan Press, 1975. — 183 p.
20. **Hoogeven J. A., van de Velde S. L., Veltman B.** Complexity of scheduling multiprocessor tasks with prespecified processors allocations // Discrete Appl. Math. — 1994. — Vol. 55. — P. 259–272.
21. **Ierapetritou M. G, Floudas C. A.** Effective continuous-time formulation for short-term scheduling: I. Multipurpose batch process // Ind. Eng. Chem. Res. — 1998. — Vol. 37. — P. 4341–4359.
22. **Itai A., Papadimitriou C. H., Szwarcfiter J. L.** Hamilton paths in grid graphs // SIAM J. Comput. — 1982. — Vol. 11, N 4. — P. 676–686.
23. **Jansen K., Porkolab L.** Preemptive scheduling with dedicated processors: applications of fractional graph coloring // J. Scheduling. — 2004. — Vol. 7, N 1. — P. 35–48.
24. **Kubale M.** Preemptive versus nonpreemptive scheduling of biprocessor tasks on dedicated processors // Eur. J. Oper. Res. — 1996. — Vol. 94. — P. 242–251.
25. **Lin X., Floudas C. A., Modi S., Juhasz N. M.** Continuous-time optimization approach for medium-range production scheduling of a multiproduct batch plant // Ind. Eng. Chem. Res. — 2002. — Vol. 41. — P. 3884–3906.
26. **Papadimitriou C. H.** Computational complexity. — Mass.: Addison Wesley, 1994. — 540 p.
27. **Reeves C. R.** Genetic algorithms for the operation researcher // IN-

FORMS J. Comput. — 1997. — Vol. 9, N 3. — P. 231–250.

28. Rudolph G. Finite Markov chain results in evolutionary computation: a tour d'horizon // Fundam. Inform. — 1998. — Vol. 35, N 1–4. — P. 67–89.

Еремеев Антон Валентинович,
e-mail: eremeev@ofim.oscsbras.ru

Коваленко Юлия Викторовна,
e-mail: juliakoval86@mail.ru

Статья поступила
17 декабря 2010 г.

Переработанный вариант —
18 мая 2011 г.