УДК 519.854.2 doi: 10.15827/0236-235X.149.027-038 2025. Т. 38. № 1. С. 27–38

Генетический алгоритм размещения требований в задаче планирования производственных процессов потокового типа

А.И. Кибзун [1], В.А. Рассказова [1]✉

[1] Институт компьютерных наук и прикладной математики,
Московский авиационный институт (национальный исследовательский университет)
(МАИ), г. Москва, 125993, Россия

Ссылка для цитирования
Кибзун А.И., Рассказова В.А. Генетический алгоритм размещения требований в задаче планирования производственных процессов потокового типа // Программные продукты и системы. 2025. Т. 38. № 1. С. 27–38. doi: 10.15827/0236-235X.149.027-038
Информация о статье
Группа специальностей ВАК: 2.3.5
Поступила в редакцию: 18.07.2024 После доработки: 14.08.2024 Принята к публикации: 23.08.2024

Аннотация. В статье рассматривается задача планирования производственных процессов потокового типа. В рамках каскадной схемы комплексное решение охватывает этап назначения подготовительных агрегатов и последующий этап формирования детализированных технологических маршрутов для исполнения заданного множества требований точно в срок и с учетом ограничений на допустимые длительности обработки на каждом переделе. Данная схема реализуется в составе проблемно-ориентированного вычислительного комплекса, однако по ряду естественных причин задача может оказаться несовместной уже на этапе назначения подготовительных агрегатов. Один из путей преодоления обозначенных трудностей – разработка и реализация алгоритмов штрафных функций для поиска максимальных совместных подсистем в противоречивых задачах оптимизации. В настоящей работе для этих целей предлагается идеологически другой подход, основанный на рассмотрении предварительного этапа размещения требований таким образом, чтобы последующие этапы решения комплексной задачи были гарантированно разрешимы. Размещение требований формализуется как задача поиска отображения установленного вида, оптимального по эвристическому критерию потенциальной нагрузки на подготовительные агрегаты в рассматриваемом периоде планирования. Для решения этой задачи авторы статьи разработали генетический алгоритм, что обусловило существенное преимущество по быстродействию в сравнении с фундаментальными подходами математического программирования (например, в сравнении с моделями целочисленного линейного программирования). В целях снижения рисков вымирания популяции на каждой итерации генетического алгоритма применяется правило безусловной миграции представителя с наименьшим значением критерия. Такой подход обеспечивает также эффективные показатели сходимости алгоритма по числу итераций без существенного улучшения целевого функционала. Разработанный генетический алгоритм реализуется как автономный модуль вычислительного комплекса для решения задач планирования процессного производства. Вычислительный эксперимент проводится с использованием данного модуля в разрезе сравнительного анализа качества решения исходной комплексной задачи.

Ключевые слова: генетический алгоритм, вычислительный комплекс, производственное планирование, процессное производство, теория расписаний, комбинаторная оптимизация

Введение. Производство потокового типа (процессное производство) представляет собой широкую прикладную область. Его отличительными особенностями являются непрерывность отдельных этапов и ритмичность операций в строго установленной последовательности. В настоящее время многие промышленные предприятия переживают стадию цифровой трансформации, когда задачи повышения эффективности производства решаются путем внедрения цифровых сервисов планирования, слежения, мониторинга и предиктивной аналитики. Такая тенденция обусловлена не только значительными оптимизационными возможностями, но и ограничениями инфраструктурных преобразований, поскольку вмешательство в топологию цехов, включая расширение парка оборудования, часто оказывается крайне трудоемким или даже невозможным. В связи с этим особую актуальность приобретают задачи разработки и внедрения полнофункциональных проблемно-ориентированных вычислительных комплексов для решения упомянутых прикладных задач.

Традиционно выделяют три уровня производственного планирования: стратегическое (верхний уровень, долгосрочное планирование), формирование комплексного графика производства (средний уровень) и оперативное планирование и управление (нижний уровень). В работе [1] обсуждаются особенности разработки и внедрения систем планирования. В настоящей статье рассматривается задача среднесрочного планирования производственных процессов потокового типа.

К современным инструментам решения задач среднесрочного планирования производственных процессов относят системы класса APS (*Advanced Planning and Scheduling*). Среди популярных отечественных систем данного класса можно выделить такие, как AVM.APS, BFG APS, БИА.APS и Аскон Гольфстрим. Из зарубежных разработок класса APS широкое распространение получила Tecnomatix Plant Simulation. Основными методами решения задач оптимизации в данных системах выступают алгоритмы имитационного моделирования и машинного обучения [2–4]. В настоящей статье для решения рассматриваемой комплексной задачи планирования используется система, в основе которой лежат подходы математического программирования [5]. Рассматриваемая комплексная задача планирования производственных процессов потокового типа включает два этапа. На первом этапе для фиксированного по времени и машинам множества требований (работ) формируется график их обработки на основных подготовительных агрегатах. На следующем этапе для фиксированных по времени, машинам и подготовительным агрегатам требований формируется детализированный технологический маршрут. Такой декомпозированный подход обусловлен практическими аспектами рассматриваемого типа производства и, кроме того, позволяет существенно снизить размерности оптимизационных задач, возникающих на каждом этапе. Однако часто уже на первом этапе решения комплексной задачи (назначение подготовительных агрегатов) возникающие оптимизационные модели оказываются несовместными. Природа такой несовместности во многом обусловлена исходным размещением требований по машинам и фиксированным временем начала и длительности их обработки. В этой связи в настоящей статье предлагается рассмотрение вспомогательного этапа размещения требований в целях обеспечения последующей гарантированной разрешимости этапов назначения подготовительных агрегатов и формирования детализированных технологических маршрутов. Для решения задачи размещения требований предлагается генетический алгоритм, реализуемый в качестве автономного модуля системы планирования [5].

Генетические алгоритмы определяют одно из фундаментальных направлений современных исследований в области случайно направленного поиска [6, 7]. Эффективность их применения для решения различных прикладных задач подтверждается многочисленными работами отечественных и зарубежных авторов. Так, например, авторы работы [8] применяют генетические алгоритмы для решения задач размещения, возникающих при проектировании интегральных схем. В [9, 10] генетические алгоритмы предложены для решения задач планирования процесса перевозок на железнодорожном транспорте. В задачах планирования в машиностроении и других серийных производствах также эффективно применяются метаэвристические подходы, в том числе генетические алгоритмы [11, 12]. Обзор различных задач производственного планирования класса RCPSP (*Resource Constrained Project Scheduling Problem*) и метаэвристические подходы к решению подробно обсуждаются в работах [13, 14]. Также широко распространены гибридные подходы на базе генетических алгоритмов. Например, в работе [15] рассматривается частный случай задачи производственного планирования потокового типа с применением гибридного алгоритма. В [16, 17] для задач с критерием минимизации простоев оборудования предложены эффективные эволюционные алгоритмы. Подходы нечеткой логики развиваются авторами [18] для решения комплексных задач производственного планирования.

Разработка нового генетического алгоритма в настоящей работе продиктована спецификой рассматриваемой задачи, когда этап размещения требований выделяется в отдельную предварительную стадию решения комплексной задачи. С этой точки зрения данная задача и методы ее решения слабо освещены в существующих публикациях.

Постановка задачи

Пусть заданы множество требований $S = \{\sigma_1, \ldots, \sigma_n\}$ и множество машин $R = \{r_1, \ldots, r_m\}$. Для каждого требования определим $\rho_i \subseteq R$ как подмножество машин, доступных для размещения σ_i. С прикладной точки зрения подмножество машин ρ_i определяется технологическими характеристиками требования σ_i. Пусть для каждого σ_i определены также τ_i, $\hat{\tau}_i$ – минимально необходимое и максимально допустимое время ожидания с момента подготовки на специализированном агрегате из множества $K = \{k_1, \ldots, k_l\}$. Допустимые минимальную и максимальную длительности обработки σ_i на машине из подмножества ρ_i обозначим соответственно через μ_i, $\hat{\mu}_i$ (рис. 1).

Определим период времени $[T_0, T]$, доступный для размещения требований S по машинам R, где T_0 принимает любое произвольное значение и

$$T = T_0 + \max_{j=1,m} \sum_{i=1}^{n} \left\{ \hat{\mu}_i \middle| r_j \in \rho_i \right\}. \qquad (1)$$

Задача размещения требований S по машинам R состоит в построении отображения вида

$$f : S \to \left\{ (r, t, m) \middle| r \in R, t, m \in \mathbb{Z}^+ \right\}, \qquad (2)$$

где $f(\sigma_i) = (r_i, t_i, m_i)$, если требование σ_i выполняется на машине r_i с началом в момент времени t_i и продолжительностью m_i, с ограничениями:

– для любого $f(\sigma_i)$ выполняется $r_i \in \rho_i$;
– для любого $f(\sigma_i)$ выполняется $T_0 - \max_{i=1,n} \{\hat{\tau}_i\} \le t_i \le T$;
– для любого $f(\sigma_i)$ выполняется $m_i \in [\mu_i, \hat{\mu}_i]$;
– для любых $f(\sigma_i)$, $f(\sigma_j) \middle| r_i = r_j$ выполняется

$$\begin{cases} t_j \ge t_i + m_i, \text{ если } t_i \le t_j, \\ t_i \ge t_j + m_j \text{ иначе.} \end{cases} \qquad (3)$$

Замечание 1. В расширенной постановке задача (2) может включать также и ограничения на длительность δ_{ij} подготовки машины $r_i = r_j$ при последовательном назначении требований $f(\sigma_i)$, $f(\sigma_j)$ (рис. 2). Авторы не рассматривают данную группу ограничений ввиду декомпозиции множества $R = \bigcup_{i=1}^{n} \rho_i$ в предположении, что длительность подготовки постоянна для требований с идентичными технологическими характеристиками и включена в интервал $[t_i, t_i + m_i]$ для любых $f(\sigma_i)$, $f(\sigma_j)$, где $r_i = r_j$ и $t_i \le t_j$.

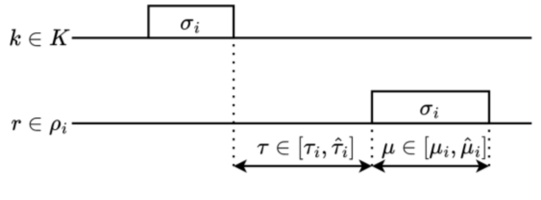

Рис. 1. Длительность ожидания и время обработки требования σ_i

Fig. 1. Waiting and processing time of requirement σ_i

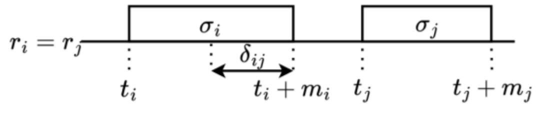

Рис. 2. Длительность подготовки при последовательном назначении требований

Fig. 2. Preparation time for sequential assignment of requirements

Замечание 2. В случае $l \le m$ задача размещения требований на горизонте $[T_0, T]$, где T определяется по правилу (1), становится тривиальной с точки зрения поиска допустимого отображения вида (2) и может быть решена жадным алгоритмом. В этой связи далее будем полагать, что $l \le \left\lceil \frac{m}{2} \right\rceil$, где

$$\left\lceil \frac{m}{2} \right\rceil = \min_{x \in \mathbb{Z}} \left\{ x \middle| x \ge \frac{m}{2} \right\}.$$

В настоящей работе в качестве критерия оптимизации размещения вида (2) рассматривается потенциальная загрузка агрегатов подготовки требований. Такой подход связан с тем, что практические задачи размещения требований, как правило, рассматриваются отдельно от задач назначения подготовительных агрегатов и последующих машин обработки. Так, в частности, время начала выполнения требования σ_i на машине r выступает входными данными и фиксировано для задачи о назначении агрегатов подготовки (рис. 3 (пунктирной линией обозначены возможные варианты назначения агрегата k для подготовки фиксированного требования σ_i)).

В работе [19] для решения задачи о назначениях подготовительных агрегатов разрабатывается модель *целочисленного линейного программирования* (ЦЛП).

Далее время завершения подготовки требования σ_i на агрегате k, а также время начала выполнения на машине r выступают входными данными и фиксированы для задачи о формировании детализированных технологических маршрутов (рис. 4 (пунктирной линией обозначены варианты назначения машин a_1 и a_2 для обработки фиксированного требования σ_i)).

Для задачи формирования детализированных технологических маршрутов в работе [20] также предлагаются каскадная схема и модели ЦЛП. Важным обстоятельством при этом является то, что уже на этапе назначения подготовительных агрегатов для фиксированного множества требований система ограничений может оказаться несовместной. Именно в этой связи, то есть для снижения рисков противоречивости входных данных, этап размещения требований выделяется в самостоятельную задачу.

Обозначим через F множество всех возможных назначений вида (2), и для каждого $f \in F$ определим множество I_f следующим образом:

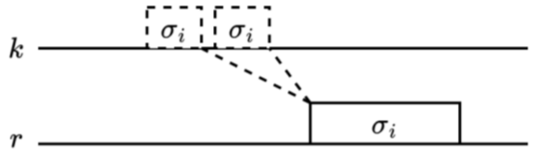

Рис. 3. Задача о назначении подготовительных агрегатов

Fig. 3. Problem of assigning preparation units

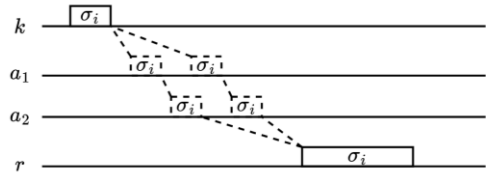

Рис. 4. Задача о формировании детализированных технологических маршрутов

Fig. 4. Problem of forming detailed technological routes

$$I_f = \bigcup_{i=1}^{n} \left\{ t_i - \hat{\tau}_i ;\ t_i - \tau_i \right\}. \tag{4}$$

Другими словами, множество I_f есть множество точек пересечения интервалов $[t_i - \hat{\tau}_i;$ $t_i - \tau_i]$, а также точек концов интервалов по всем требованиям $\sigma_i \in S$. Например, для $S = \{\sigma_1, \sigma_2\}$ и $f(\sigma_1)$, $f(\sigma_2)$, таких, что $[t_2 - \hat{\tau}_2; t_2 - \tau_2] \subset [t_1 - \hat{\tau}_1;\ t_1 - \tau_1]$, множество I_f примет вид

$$I_f = \{ t_1 - \hat{\tau}_1,\ t_2 - \hat{\tau}_2,\ t_2 - \tau_2,\ t_1 - \tau_1 \}.$$

Пусть длительность подготовки любого требования $\sigma_i \in S$ постоянна для всех агрегатов $k_1, \ldots, k_l$ и равна λ. Для любых $f \in F$ и $d^k \in I_f$, $k = \overline{1, |I_f| - 1}$, определим величину

$$L(d^k, f) = \frac{\left| d^{k+1} - d^k \right| l}{\lambda \left| \left\{ f(\sigma_i) : d^k \le t_i < d^{k+1} \right\} \right|} \tag{5}$$

и критерий

$$P(f) = \max_{k = \overline{1, |I_f - 1|}} \left\{ L(d^k, f) \right\}. \tag{6}$$

Величина (5) по сути отражает количество требований, потенциально задействующих каждый из l агрегатов подготовки в момент времени d^k. При этом ясно, что внутри интервала $[d^k, d^{k+1}]$ значение (5) не меняется, чем и обусловлено его вычисление только в точках из множества I_f. Таким образом, для заданных S, R, K и $[T_0, T]$ задача размещения требований S по машинам R может быть сведена к оптимизационной задаче поиска отображения вида (2), такого, что

$$P(f) \to \min_{f \in F}. \tag{7}$$

Для решения задач (2) и (7) разработан генетический алгоритм.

Генетический алгоритм размещения требований

Идеологически генетический алгоритм включает в себя этапы создания начальной популяции, селекции, скрещивания и мутации.

Пусть заданы множество требований $S = \{\sigma_1, \ldots, \sigma_n\}$, период планирования $[T_0, T]$ и множество подготовительных агрегатов $K = \{k_1, \ldots, k_l\}$ с фиксированной длительностью $\lambda(k_1) = \ldots = \lambda(k_l) = \lambda$ для подготовки любого требования $\sigma_i \in S$. Под особью, или представителем популяции, будем понимать назначение вида $f(S) = \{f(\sigma_1), \ldots, f(\sigma_n)\}$. При этом различными генами особи $f(S)$ являются назначения вида $f(\sigma_i) = (r_i, t_i, m_i)$. Обозначим через F^i популяцию для i-й итерации алгоритма, где

$F^i = \{f_1(S), f_2(S), \ldots\}$.

Определим следующий набор параметров генетического алгоритма:

– размерность K_1 для начальной популяции F^0;

– отбор K_2 представителей популяции в порядке возрастания значения критерия (6);

– формирование K_3 потомков для каждой пары отобранных представителей;

– мутация K_4 генов;

– ограничение τ (мин.) для процедуры мутации генов;

– критерий останова по совокупному числу K_5 эпох;

– критерий ε (%) улучшения целевого функционала для последовательных итераций алгоритма;

– критерий останова по числу K_6 эпох (итераций алгоритма) без улучшения целевого функционала.

Далее при описании алгоритмов после символа // дается комментарий к соответствующей строке.

Генетический алгоритм размещения требований:

1. Выполнить процедуру формирования начальной популяции F^0 из K_1 особей

2. $k = 0$, $c = 0$ // счетчики совокупного числа эпох и числа эпох без улучшения целевого функционала

3. Выполнить процедуру отсечения популяции F^0

4. **Пока** $k \neq K_5$ или $c \neq K_6$

5. Выполнить процедуру отбора K_2 представителей текущей популяции F^k в порядке возрастания критерия (6)

6. Выполнить процедуру скрещивания всех $\frac{K_2 \cdot (K_2 - 1)}{2}$ пар представителей, отобранных на шаге 5 // увеличить счетчик числа эпох, включить в новую популяцию лучшего представителя текущей и сформировать K_3 представителей-потомков для каждой пары

7. Выполнить процедуру мутации генов

8. Выполнить процедуру отсечения текущей популяции F^k

9. **Вернуть** $f : P(f) \to \min\limits_{f \in F^k}$ // наилучший по критерию (6) представитель текущей популяции

10. **Если** минимальное значение критерия (6) на шаге 5 отклоняется от значения на шаге 9 менее, чем на ε (%), то

11. Увеличить счетчик числа эпох без улучшения целевого функционала

Рассмотрим подробнее этапы данного генетического алгоритма.

Процедуры формирования начальной популяции и отсечения

Для формирования начальной популяции F^0 размещение $f(S)$ требований производится случайным образом посредством выбора некоторых значений для машины, времени начала и длительности обработки из числа допустимых. Далее будем полагать, что время T_0, T определяются целым числом в формате timestamp (ts). Время t_i начала обработки каждого требования также будем определять в формате ts, а длительность обработки m_i – в формате целого числа (количество минут).

Алгоритм 1. Процедура формирования начальной популяции

1. $F^0 = \{\}$

2. **Для всех** $k = \overline{1, n}$ // количество особей $n = K_1$

3. $f_k(S) = \{\}$

4. **Для всех** $\sigma_i \in S$

5. $t_i = Rnd(T_0, T)$ // случайное число в диапазоне от T_0 до T

6. $m_i = Rnd(\mu_i, \hat{\mu}_i)$

7. $r_i = Rnd(1, |\rho_i|)$

8. $f(\sigma_i) = (r_i, t_i, m_i)$

9. $f_k(S) = f_k(S) + \{f(\sigma_i)\}$ // сформировать ген $f(\sigma_i)$ для текущей особи $f_k(S)$

10. $F^0 = F^0 + \{f_k(S)\}$

11. **Вернуть** $F^0 = \{f_1(S), \ldots, f_n(S)\}$

Фрагмент начальной популяции из двух особей с пятью генами отображен на рисунке 5. Например, для требования σ_1 на итерации $k = 1$ внешнего цикла алгоритма 1 установлена машина обработки $r_1 = r^1$, а на итерации $k = 2$ машина $r_1 = r^2$. Аналогично для итераций $k = 1$ и $k = 2$ на рисунке 5 представлены различные значения для t_1 и m_1 и т.д. для всех рассматриваемых требований σ_i, $i = \overline{1, 5}$.

Таким образом, в строках 5–7 алгоритма 1 для каждого требования $\sigma_i \in S$ выбираются произвольные время начала $t_1 \in [T_0, T]$, длительность $m_i \in [\mu_i, \hat{\mu}_i]$ и машина обработки $r_1 \in \rho_i$. При этом ясно, что некоторые особи $f(S) \in F^0$ могут оказаться противоречивыми с точки зрения ограничений (3) задачи (2) (время начала и длительность обработки требований σ_i, σ_j при их последовательном размещении на одну и ту же машину $r_i = r_j$). На рисунке 6 представлен пример особи $f(S)$ с противоречивыми генами $f(\sigma_1)$ и $f(\sigma_2)$, такими, что $r_1 = r_2$, $t_1 < t_2$ и $t_1 + m_1 > t_2$.

Процедура отсечения популяции выполняется для начальной популяции и на каждой итерации внешнего цикла генетического алгоритма, обеспечивая тем самым рассмотрение только допустимых назначений вида (2). При этом размерность текущей популяции после

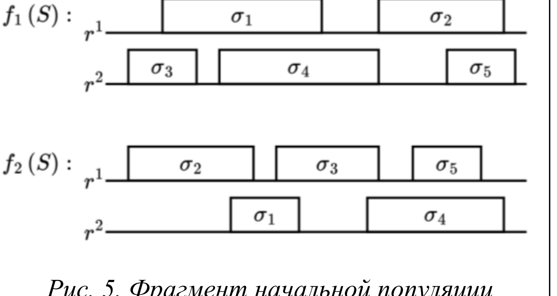

Рис. 5. Фрагмент начальной популяции

Fig. 5. Initial population fragment

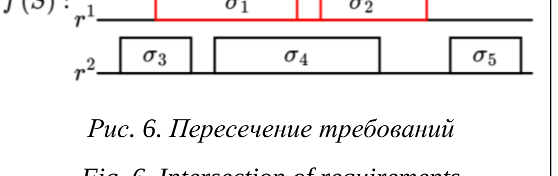

Рис. 6. Пересечение требований

Fig. 6. Intersection of requirements

выполнения процедуры отсечения во многом определяется значениями параметров K_1 и K_2 алгоритма. Так, в случае тенденции вымирания соответствующие параметры могут быть существенно увеличены в целях стабилизации размерности.

Алгоритм 2. Процедура отсечения популяции

1. **Для всех** $f(S) \in F^k$
2. **Для всех** $f(\sigma_i) \in f(S)$ // по всем парам различных генов
3. **Для всех** $f(\sigma_j) \in f(S) \big| \sigma_i \neq \sigma_j$
4. **Если** $r_i = r_j$ **и** $t_j \le t_i$ и $t_i + m_j > t_i$ **то** // ограничения (3) нарушены
5. $F^k = F^k - \{ f(S) \}$ // исключить особь из текущей популяции
6. Выход из цикла
7. **Если** $r_i = r_j$ и $t_i < t_j$ и $t_j + m_j > t_i$ **то**
8. $F^k = F^k - \{ f(S) \}$
9. Выход из цикла
10. Вернуть F^k

Процедура отбора представителей

Рассмотрим начальную популяцию:

$$F^0 = \{ f_1(S), f_2(S), \ldots \},$$

где

$f_j(S) = \{ f_j(\sigma_1), f_j(\sigma_2), \ldots \}$ для всех j = 1, 2, ...

и

$f_j(\sigma_i) = (r_i, t_i, m_i)$ для всех $\sigma_i \in S$.

Напомним, что любое размещение требований $f_j(S) \in F^0$ является допустимым с точки зрения ограничений задачи (2) ввиду структуры алгоритма 1 и с учетом выполнения процедуры отсечения согласно алгоритму 2.

Для каждого представителя $f_j(S) \in F^0$ определим множество (4) и значение критерия (6) при фиксированных l и λ для рассматриваемого количества подготовительных агрегатов и длительности подготовки каждого требования соответственно. На рисунке 7 приведен пример множества I_f для некоторого представителя $f(S)$ с пятью генами $f(\sigma_i), i = \overline{1,5}$ – точки на верхней горизонтальной прямой.

Аналогично множество I_f и значение критерия (6) могут быть определены для представителей любой популяции F^k (условие допустимости всех представителей $f_j(S)$ произвольной популяции F^k с точки зрения ограничений задачи (2) будут обоснованы несколько позже на этапе обсуждения процедуры мутации генов).

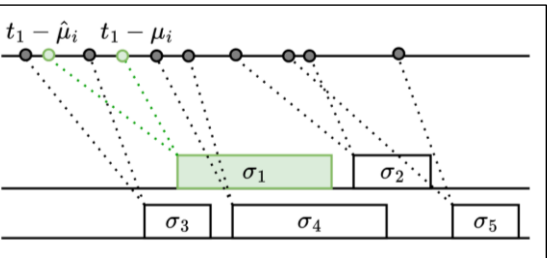

Рис. 7. Множество I_f

Fig. 7. Set I_f

Алгоритм 3. Процедура отбора представителей

1. **Для всех** $f_j(S) \in F^k$
2. $f = f_j(S)$
3. Сформировать множество точек $I_f = \{ d^1, d^2, \ldots \}$
4. **Для всех** $d^n \in I_f$ // $n = \overline{1, |I_f| - 1}$
5. $L(d^n, f) = \dfrac{|d^{k+1} - d^k| l}{\lambda \left| \{ f(\sigma_i) \in f : d^k \le t_i < d^{k+1} \} \right|}$
6. $L_j = \max\limits_n \{ L(d^n, f) \}$
7. **Вернуть** $\{L_1, L_2, \ldots\}$
8. Отсортировать F^k в порядке возрастания значений L_j ее представителей: $\{ f_{j_1}(S), f_{j_2}(S), \ldots \}$, где $L_{j_1} \le L_{j_2} \le \ldots$
9. **Вернуть** $F^k = \{ f_{j_1}(S), \ldots, f_{j_n}(S) \}$ // $n = K_2$
10. **Вернуть** $f^* = f_{i_1}(S)$ // наилучший представитель текущей популяции

Процедуры скрещивания и мутации

Рассмотрим популяцию $F^k = \{ f_{j_1}(S), \ldots, f_{j_n}(S) \}$, построенную в результате выполнения алгоритма 3 как подмножество $n = K_2$ лучших представителей по критерию (6). Для каждой пары представителей $f_i(S)$, $f_j(S) \in F^k$ гены представителя-потомка $f(S)$ определяются посредством прямого наследования от случайным образом выбранного родителя $f_i(S)$ или $f_j(S)$. На рисунке 8 приведен пример формирования представителя потомка $f(S)$ для родителей $f_1(S)$, $f_2(S)$ с генами $f(\sigma_i), i = \overline{1,5}$, где гены $f(\sigma_i), i = \overline{1,3}$, наследуются от родителя $f_1(S)$ и гены $f(\sigma_i), i = \overline{4,5}$, от родителя $f_2(S)$ соответственно.

Алгоритм 4. Процедура скрещивания

1. $F = F^k, k = k + 1, F^k = \{ f^* \}$
2. **Для всех** $f_i(S) \in F$, $i = \overline{1,n}$ // $n = K_2$

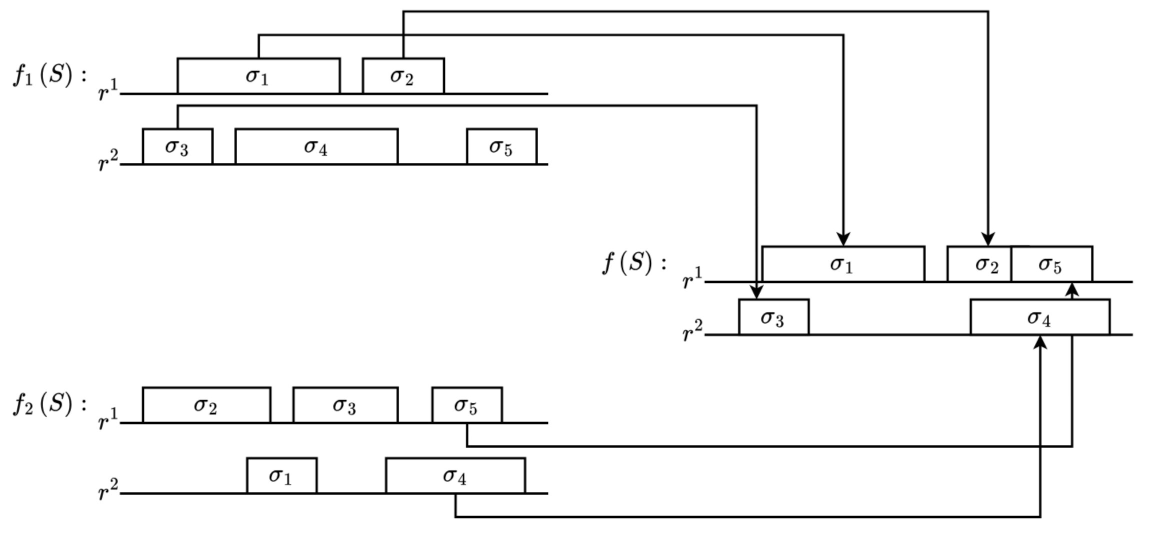

Рис. 8. Представитель-потомок f(S)

Fig. 8. Descendant representative f(S)

3. **Для всех** $f_j(S) \in F$, $j = \overline{i+1,n}$ // для всех пар родителей

4. **Для всех** $l = \overline{1,m}$ // сформировать $m = K_3$ потомков

5. $f_l(S) = \{\}$

6. **Для всех** $\sigma \in S$

7. $p = Rnd(0,1)$

8. **Если** $p \leq 0.5$ то

9. $f(\sigma) = f_i(\sigma)$ // наследовать текущий ген от родителя $f_i(S)$

10. Иначе

11. $f(\sigma) = f_j(\sigma)$

12. $f_l(S) = f_l(S) + \{f(\sigma)\}$

13. $F^k = F^k + \{f_l(S)\}$

14. Вернуть F^k, k

На шаге 1 алгоритма 4 формируется новая популяция F^k, в состав которой включается наилучший по критерию (6) представитель f^* предшествующей популяции F. Данная процедура обеспечивает сохранение на каждой итерации текущего локально оптимального решения и последующий контроль числа K_6 итераций без существенного улучшения целевого функционала – параметр ε (%) алгоритма.

Для каждого представителя текущей популяции $f(S) \in F^k$ случайным образом выбранные K_4 генов $f(\sigma)$ мутируют в части времени начала и длительности исполнения соответствующего требования $\sigma \in S$ по процедуре, описанной в алгоритме 1.

Алгоритм 5. Процедура мутации генов

1. **Для всех** $f(S) \in F^k$

2. **Для всех** $i = \overline{1,n}$ // мутация $n = K_4$ случайно выбранных генов

3. $j = Rnd(1,|S|)$ // выбор произвольного $\sigma \in S$

4. $t_j = t_j + Rnd(-\tau, \tau)$

5. $f(\sigma_j) = (r_j, t_j, m_j)$

6. **Вернуть** F^k

Аналогично алгоритмам 1 и 2 структура алгоритма 5 и последующий вызов процедуры отсечения в строке 8 генетического алгоритма обеспечивают допустимость любого размещения требований $f(S) \in F^k$ с точки зрения ограничений задачи (2).

Вычислительный эксперимент

Предложенный генетический алгоритм размещения требований, включая процедуры, описанные в алгоритмах 1–5, были реализованы на языке Python 3.8 как автономный модуль комплексной системы планирования производственных процессов потокового типа [2]. Анализ эффективности применения генетического алгоритма на предварительном этапе решения комплексной задачи проводится в двух направлениях: с точек зрения оптимизации входных данных для задачи о назначении подготовительных агрегатов (в случае ее исходной противоречивости) и трудоемкости этой задачи (в случае наличия решения для исходной постановки). Общая структура вычислительного эксперимента представлена на рисунке 9.

Вычислительный эксперимент был проведен на реальных производственных данных. Рассматриваются годовой период и множество требований $S = \{\sigma_1, \ldots, \sigma_n\}$, подлежащих исполнению в каждые сутки рассматриваемого периода. Таким образом, для каждого экземпляра входных данных период планирования $[T_0, T]$ составляет 24 часа. Множество S содержит $n = 100$ требований с параметрами:

– $30 \le \tau_i < \hat{\tau}_i \le 180$ (мин.),

– $30 \le \mu_i < \hat{\mu}_i \le 60$ (мин.),

– $\rho_i = R$

для всех $i = \overline{1{,}100}$, где $R = \{r_j\}, j = \overline{1{,}5}$ – множество машин, доступных для исполнения требований множества S. Для множества подготовительных агрегатов $K = \{k_l\}, l = \overline{1{,}3}$, длительность обработки каждого требования фиксированная и составляет $\lambda = 40$ (мин.).

Определим параметры генетического алгоритма:

– размерность начальной популяции $K_1 = 1\,000$ представителей;

– отбор $K_2 = 30$ представителей популяции в порядке возрастания значения критерия (6);

– формирование $K_3 = 50$ потомков для каждой пары отобранных представителей;

– мутация $K_4 = 2$ гена;

– ограничение $\tau = 5$ (мин.) для процедуры мутации генов;

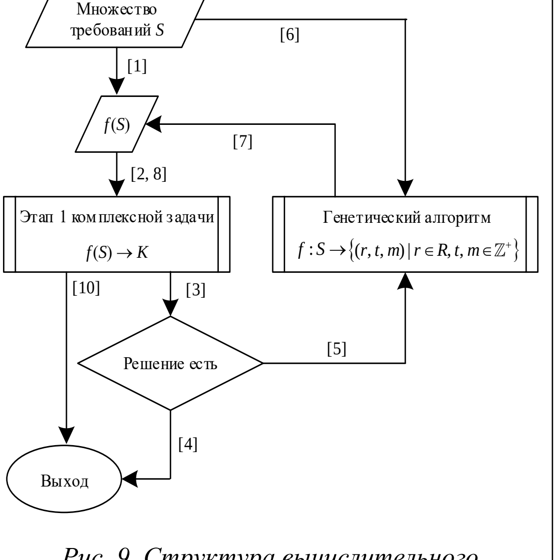

Рис. 9. Структура вычислительного эксперимента

Fig. 9. Computational experiment structure

– критерий останова по числу $K_5 = 10$ эпох (итераций алгоритма) без улучшения целевого функционала;

– критерий $\varepsilon = 10$ (%) улучшения целевого функционала для последовательных итераций алгоритма;

– критерий останова по совокупному числу $K_6 = 500$ эпох.

Замечание 3. Если размерность популяции на некоторой итерации алгоритма не превышает значение параметра $K_2 = 30$, то процедура формирования $K_3 = 50$ потомков производится для каждой пары представителей соответствующей популяции.

На рисунке 10 показано соотношение величины [5] в каждой точке множества I_f соответственно для размещения, сформированного посредством жадного алгоритма (см. поток [1] на рисунке 9), и для размещения, построеннного с использованием генетического алгоритма (см. потоки [6] и [7] на рисунке 9).

В среднем значение величины $L(d^k, f)$ (потенциальная нагрузка на подготовительные агрегаты) составляет 0,6 в обоих случаях. Однако разброс значений по всем точкам $d^k \in I_f$ для размещения, построенного с использованием генетического алгоритма, существенно меньше и равен [0,35;0,83] по сравнению с [0,3;1,39] соответственно для размещения, сформированного посредством жадного алгоритма.

В примере на графике значение максимума потенциальной нагрузки на подготовительные агрегаты (6) составляет 1,39 и 0,83 для жадного и генетического алгоритмов соответственно. Аналогичные расчеты были проведены для каждого экземпляра входных данных в рассматриваемом годовом периоде. Результаты сравнительного анализа по критерию (6) представлены на графике (http://www.swsys.ru/uploaded/image/2025-1/25.jpg).

Диапазон значений критерия (6) в случае размещения, сформированного посредством жадного алгоритма, составляет [1,01;2,8] со средним значением 1,91. В то же время для размещения, построенного с использованием генетического алгоритма, диапазон значений критерия (6) составляет [0,8;1,27] со средним значением 1,07.

На последующем этапе назначения подготовительных агрегатов (см. потоки [2] и [8] на рисунке 9) фиксировалось время в минутах, затраченное системой на поиск решения. Для случаев, когда решение не было найдено, данный показатель принимался равным 0 (см. потоки [3] и [5] на рисунке 9). Проведен анализ

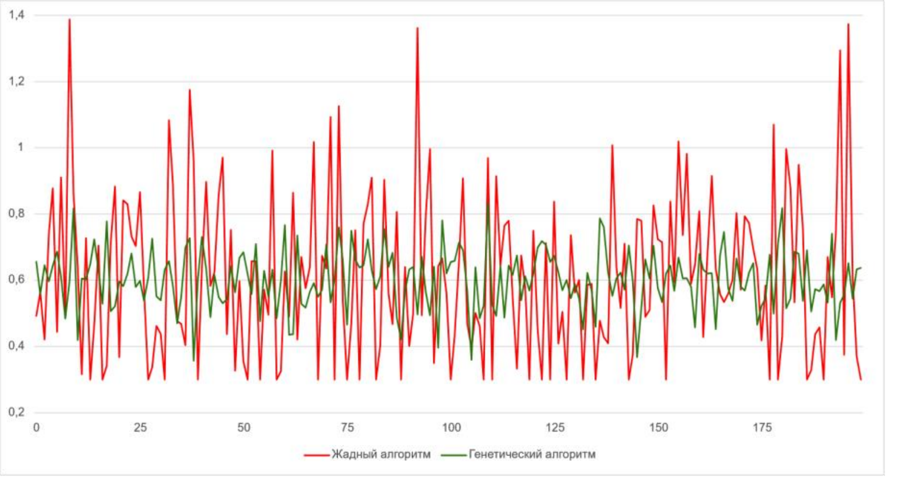

Рис. 10. Значение $L(d^k, f)$ для точек множества I_f

Fig. 10. The value of $L(d^k, f)$ for the points of set I_f

данного показателя временных затрат для каждого экземпляра входных данных в рассматриваемом годовом периоде (http://www.swsys.ru/uploaded/image/2025-1/26.jpg).

Можно сделать вывод, что решения нет во всех случаях, когда значение критерия (6) превышает 2,1. Доля таких случаев для размещения, построенного с использованием жадного алгоритма, составляет 40 %, а для размещения, сформированного посредством генетического алгоритма, значение критерия (6) не превышает 1,27 в 100 % случаев. В 60 % случаев, когда решение было найдено для размещения, построенного по жадному алгоритму, время поиска решения варьируется в диапазоне [2,01;6,96] со средним значением 2,71 мин. Для размещения, построенного по генетическому алгоритму, диапазон времени поиска решения на этапе назначения подготовительных агрегатов составляет [1;2,92] со средним значением 1,97 мин.

Из результатов проведенного вычислительного эксперимента следует, что применение генетического алгоритма на этапе размещения требований существенно снижает критерий максимальной потенциальной нагрузки на подготовительные агрегаты по сравнению с решением данного этапа посредством жадного алгоритма. Последующий этап назначения подготовительных агрегатов оказывается разрешимым в 100 % случаев в рассматриваемом годовом периоде, в то время как для жадного размещения в 40 % случаев решение не найдено. Наконец, сравнительный анализ временных затрат демонстрирует потенциал ускорения в среднем до 27 % в случаях, когда для жадного размещения найдено решение на этапе назначения подготовительных агрегатов.

Заключение

В статье рассмотрена задача размещения требований как начальный этап комплексного планирования производственных процессов потокового типа. Такая декомпозиция обусловлена прикладными аспектами комплексной задачи, а также существенным потенциалом для снижения исходной размерности. Кроме того, рассмотрение вспомогательного этапа размещения требований позволяет ожидать разрешимости последующих этапов в случае противоречивости ограничений. Для задачи размещения требований в рассмотрение введен эвристический критерий потенциальной нагрузки на подготовительные агрегаты и предложен генетический алгоритм решения. Подробно обсуждаются ключевые процедуры алгоритма. Предложенный генетический алгоритм реализуется на языке Python как автономный модуль системы планирования производственных процессов потокового типа. Вычислительный эксперимент с использованием этой системы проводится на реальных производственных данных по двум направлениям: с точек зрения разрешимости последующего этапа назначения подготовительных агрегатов и трудоемкости поиска решения для случаев размещения требований с использованием жадного подхода и генетического алгоритма соответственно. Результаты вычислительного эксперимента демонстрируют высокую эффективность генетического алгоритма по обоим аспектам сравнительного анализа. В частности, этап назначения подготовительных агрегатов оказывается разрешимым в 100 % случаев с использованием генетического алгоритма для предварительного размещения требований. С точки зрения вычислительной трудоемкости использование генетического алгоритма в проведенном эксперименте влечет ускорение до 27 % в среднем.

Дальнейшее развитие полученных результатов связано с исследованием теоретических вопросов сходимости и устойчивости предложенного генетического алгоритма. Важным этапом при этом выступает проведение масштабного вычислительного эксперимента с использованием данных тестовых библиотек. Также ставится задача разработки и реализации дополнительных оригинальных тестовых задач, позволяющих адекватно оценить поведение и свойства алгоритма. С практической точки зрения направление дальнейшего развития связано с исследованием возможностей применимости предложенного метаэвристического подхода для решения задачи планирования производственных процессов потокового типа на других этапах технологической цепочки.

Список литературы

1. Ризванов Д.А., Чернышев Е.С. Информационное и алгоритмическое обеспечение планирования производственных мощностей // Интеллектуальные системы в производстве. 2020. Т. 18. № 4. С. 117–125. doi: 10.22213/2410-9304-2020-4-117-125.

2. Махитько В.П., Рамзаев В.М., Гришанов Г.М. Экономические аспекты имитационного моделирования производственных участков в среде Technomatix Plant Simulation // Вестн. Самарского муниципального института управления. 2019. № 3. С. 7–16.

3. Никишечкин П.А., Ивашин С.С., Черненко В.Е. и др. Система имитационного моделирования PlantTwin как инструмент верификации производственных планов и поддержки принятия решений для повышения эффективности производства // Вестн. машиностроения. 2021. № 3. С. 80–85. doi: 10.36652/0042-4633-2021-3-80-85.

4. Александров В.Р., Баранов С.Е., Кузнецов М.И. и др. Искусственный интеллект в задачах планирования производства // Инфокоммуникационные и радиоэлектронные технологии. 2022. Т. 5. № 2. С. 196–208. doi: 10.29039/2587-9936.2022.05.2.14.

5. Гайнанов Д.Н., Беренов Д.А., Рассказова В.А. и др. Data-PLAN: Свид. о регистр. ПрЭВМ № 2021665276. Рос. Федерация, 2021.

6. Частикова В.А., Чич А.И. Генетические алгоритмы и генетическое программирование: особенности реализации // Перспективы науки. 2019. № 1. С. 13–16.

7. Минитаева А.М., Векшин Р.Д., Шатилов А.А. Анализ различных видов генетических алгоритмов в задачах оптимизации // Технологии инженерных и информационных систем. 2022. № 1. С. 21–34.

8. Курейчик В.М., Данильченко В.И. Генетический алгоритм планирования размещения СБИС // Изв. ЮФУ. Технич. науки. 2019. № 2. С. 26–34.

9. Суслов Д.А., Сенченко К.А., Шмаль В.Н. Применение генетических алгоритмов в сфере организации пассажирских перевозок // Дневник науки. 2022. № 9. URL: https://dnevniknauki.ru/images/publications/2022/9/technics/Suslov_Senchenko_Shmal.pdf (дата обращения: 23.07.2024).

10. Сидоренко В.Г., Сафронов А.И. Применение генетических алгоритмов при решении задач планирования перевозочного процесса городской рельсовой транспортной системы // Автоматика на транспорте. 2023. Т. 9. № 1. С. 49–62. doi: 10.20295/2412-9186-2023-9-01-49-62.

11. Гусев П.Ю., Гусев К.Ю., Вахмин С.Ю. Применение генетических алгоритмов в оптимизации планировочных решений производственных подразделений машиностроительных предприятий // Вестн. ВГТУ. 2019. Т. 15. № 2. С. 22–28.

12. Семенов Г.Е., Кейно П.П. Применение математических моделей на основе генетических алгоритмов в задачах планирования сложных технических объектов // Прикладная информатика. 2019. Т. 14. № 2. С. 56–62.

13. Liang Z., Zhong P., Liu M., Zhang Ch., Zhang Z. A computational efficient optimization of flow shop scheduling problems. Sci. Reports, 2022, vol. 12, art. 845. doi: 10.1038/s41598-022-04887-8.

14. Türkakın O.H., Arditi D., Manisali E. Comparison of heuristic priority rules in the solution of the resource-constrained project scheduling problem. Sustainability, 2021, vol. 13, no. 17, art. 9956. doi: 10.3390/su13179956.

15. Abdel-Basset M., Manogaran G., El-Shahat D., Mirjalili S. A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem. FGCS, 2018, vol. 85, pp. 129–145. doi: 10.1016/j.future.2018.03.020.

16. Kadarkarainadar M.M., Tosun Ö., Geetha M. Hybrid monkey search algorithm for flow shop scheduling problem under makespan and total flow time. Applied Soft Computing J., 2017, vol. 55, pp. 82–92. doi: 10.1016/j.asoc.2017.02.003.

17. Гончаров Е.Н. Алгоритм локального поиска для задачи календарного планирования с ограниченными ресурсами // Дискретный анализ и исследование операций. 2022. Т. 29. № 4. С. 15–37.

18. Ladj A., Tayeb F.B.-S., Varnier C. Hybrid of metaheuristic approaches and fuzzy logic for the integrated flowshop scheduling with predictive maintenance problem under uncertainties. EJIE, 2021, vol. 15, no. 5, pp. 675–710. doi: 10.1504/EJIE.2021.117325.

19. Rasskazova V.A. LIP model in solving RCPSP at the flow type production. In: CCIS. Proc. OPTIMA, 2023, vol. 1913, pp. 75–88. doi: 10.1007/978-3-031-48751-4_6.

20. Kibzun A.I., Rasskazova V.A. LIP model as mathematical ware for an optimal flow production planning system at operational scheduling stage. Automation and Remote Control, 2023, vol. 84, no. 5, pp. 529–542. doi: 10.1134/S0005117923050065.

Software & Systems doi: 10.15827/0236-235X.149.027-038 2025, 38(1), pp. 27–38

Genetic algorithm for placing requirements in a flow-type production process planning problem

Andrey I. Kibzun [1], Varvara A. Rasskazova [1]✉

[1] Institute of Computer Sciences and Applied Mathematics, Moscow Aviation Institute (National Research University) (MAI), Moscow, 125993, Russian Federation

For citation

Kibzun, A.I., Rasskazova, V.A. (2025) ‘Genetic algorithm for placing requirements in a flow-type production process planning problem’, *Software & Systems*, 38(1), pp. 27–38 (in Russ.). doi: 10.15827/0236-235X.149.027-038

Article info

Received: 18.07.2024 After revision: 14.08.2024 Accepted: 23.08.2024

Abstract. The paper discusses the problem of planning flow-type production processes. In terms of a cascade scheme, the complex solution covers the stage of assigning preparatory units and the subsequent stage of forming detailed technological routes to fulfill a given set of requirements on time and taking into account the constraints on permissible processing durations at each processing stage. This scheme comes as a part of a problem-oriented computing complex. However, due to a number of natural reasons, the problem may become inconsistent right at the stage of assigning preparatory units. One of the ways to overcome these difficulties is to develop and implement penalty function algorithms to find the maximum joint subsystems in inconsistent optimization problems. The paper proposes an ideologically different approach for this purpose. It is based on considering the preliminary stage of requirement placement in such a way that the subsequent stages of problem-solving process are guaranteed to be solvable. The requirement placement is formalized as a search for an optimal mapping that minimizes “potential” workload on preparatory units during the planning period. To solve this problem, the authors of the paper have developed a genetic algorithm, which resulted in a significant advantage in terms of speed in comparison with fundamental approaches of mathematical programming (for example, integer linear programming models). In order to reduce the risk of population extinction at each iteration of the genetic algorithm, the authors apply the rule of unconditional migration of a representative with the lowest criterion value. This approach also provides effective convergence indices of the algorithm in terms of the number of iterations without significant improvement of the objective function. The developed genetic algorithm is implemented as a stand-alone module of a computing system for solving process manufacturing scheduling problems. The authors conducted a computational experiment using this module in terms of a comparative analysis of the solution quality of the initial complex problem.

Keywords: genetic algorithm, computing complex, manufacturing planning, process manufacturing, schedule theory, combinatorial optimization

References

1. Rizvanov, D.A., Chernyshev, E.S. (2020) ‘Information and algorithmic support of production capacity planning’, *Intellekt. Sist. Proizv.*, 18(4), pp. 117–125 (in Russ.). doi: 10.22213/2410-9304-2020-4-117-125.

2. Makhitko, V.P., Ramzaev, V.M., Grishanov, G.M. (2019) ‘Economic aspects of simulation of production sites in the Technomatix Plant Simulation environment’, *Bull. of the Samara Municipal Institute of Management*, (3), pp. 7–16 (in Russ.).

3. Nikishechkin, P.A., Ivashin, S.S., Chernenko, V.E., Malyihanov, A.A., Dolgov, N.V. (2021) ‘PlantTwin simulation system as a tool for verifying production plans and supporting the decision-making to improve production effectiveness’, *Bull. of Mechanical Engineering*, (3), pp. 80–85 (in Russ.). doi: 10.36652/0042-4633-2021-3-80-85.

4. Aleksandrov, V.R., Baranov, S.E., Kuznetsov, M.I. et al. (2022) ‘Artificial intelligence in the tasks of manufacturing planning’, *Infocommunication and Radio Tech.*, 5(2), pp. 196–208 (in Russ.). doi: 10.29039/2587-9936.2022.05.2.14.

5. Gainanov, D.N., Berenov, D.A., Rasskazova, V.A. et al. (2021) *Data-PLAN*, Pat. RF, № 2021665276.

6. Chastikova, V.A., Chich, A.I. (2019) ‘Genetic algorithms and genetic programming: features of realization’, *Sci. Prospects*, (1), pp. 13–16 (in Russ.).

7. Minitaeva, A.M., Vekshin, R.D., Shatilov, A.A. (2022) ‘Analysis of various types of genetic algorithms in optimization problems’, *Tech. of Eng. and Inform. Sys.*, (1), pp. 21–34 (in Russ.).

8. Kureychik, V.M., Danilchenko, V.I. (2019) ‘Genetic algorithm of VLSI placement planning’, *Izv. SFedU. Eng. Sci.*, (2), pp. 26–34 (in Russ.).

9. Suslov, D.A., Senchenko, K.A., Shmal, V.N. (2022) ‘Application of genetic algorithms in the field of organizing passenger transportation’, *Diary of Sci.*, (9), available at: https://dnevniknauki.ru/images/publications/2022/9/technics/Suslov_Senchenko_Shmal.pdf (accessed July 23, 2024) (in Russ.).

10. Sidorenko, V.G., Safronov, A.I. (2023) ‘Application of genetic algorithms at solution of tasks for transportation process planning of city rail transport system’, *Transport Automation Research*, 9(1), pp. 49–62 (in Russ.). doi: 10.20295/2412-9186-2023-9-01-49-62.

11. Gusev, P.Yu., Gusev, K.Yu., Vakhmin, S.Yu. (2019) ‘Modeling of the dissolution and growth of sugar crystals’, *Bull. of VSTU*, 15(2), pp. 22–28 (in Russ.).

12. Semenov, G.E., Keino, P.P. (2019) ‘Mathematical model of genetic algorithm in implementation for scheduling tasks of complex technical objects during pre-production stages’, *Applied Inform.*, 14(2), pp. 56–62 (in Russ.).

13. Liang, Z., Zhong, P., Liu, M., Zhang, Ch., Zhang, Z. (2022) ‘A computational efficient optimization of flow shop scheduling problems’, *Sci. Reports*, 12, art. 845. doi: 10.1038/s41598-022-04887-8.

14. Türkakın, O.H., Arditi, D., Manisali, E. (2021) ‘Comparison of heuristic priority rules in the solution of the resource-constrained project scheduling problem’, *Sustainability*, 13(17), art. 9956. doi: 10.3390/su13179956.

15. Abdel-Basset, M., Manogaran, G., El-Shahat, D., Mirjalili, S. (2018) ‘A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem’, *FGCS*, 85, pp. 129–145. doi: 10.1016/j.future.2018.03.020.

16. Kadarkarainadar, M.M., Tosun, Ö., Geetha, M. (2017) ‘Hybrid monkey search algorithm for flow shop scheduling problem under makespan and total flow time’, *Applied Soft Computing J.*, 55, pp. 82–92. doi: 10.1016/j.asoc.2017.02.003.

17. Goncharov, E.N. (2022) ‘A local search algorithm for a scheduling problem with limited resources’, *Discrete Analysis and Operations Research*, 29(4), pp. 15–37 (in Russ.).

18. Ladj, A., Tayeb, F.B.-S., Varnier, C. (2021) ‘Hybrid of metaheuristic approaches and fuzzy logic for the integrated flowshop scheduling with predictive maintenance problem under uncertainties’, *EJIE*, 15(5), pp. 675–710. doi: 10.1504/EJIE.2021.117325.

19. Rasskazova, V.A. (2023) ‘LIP model in solving RCPSP at the flow type production’, in *CCIS. Proc. OPTIMA*, 1913, pp. 75–88. doi: 10.1007/978-3-031-48751-4_6.

20. Kibzun, A.I., Rasskazova, V.A., (2023) ‘LIP model as mathematical ware for an optimal flow production planning system at operational scheduling stage’, *Automation and Remote Control*, 84(5), pp. 529–542. doi: 10.1134/S0005117923050065.

Авторы
Кибзун Андрей Иванович [1],
д.ф.-м.н., профессор,
заведующий кафедрой,
kibzun@mail.ru
Рассказова Варвара Андреевна [1],
к.ф.-м.н., доцент,
varvara.rasskazova@mail.ru

[1] Институт компьютерных наук и прикладной математики, Московский авиационный институт (национальный исследовательский университет) (МАИ), г. Москва, 125993, Россия

Authors
Andrey I. Kibzun [1],
Dr.Sci. (Physics and Mathematics),
Professor, Head of Chair,
kibzun@mail.ru
Varvara A. Rasskazova [1],
Cand. of Sci. (Physics and Mathematics),
Associate Professor, varvara.rasskazova@mail.ru

[1] Institute of Computer Sciences and Applied Mathematics, Moscow Aviation Institute (National Research University) (MAI), Moscow, 125993, Russian Federation