УДК 519.85:656.7 DOI: http://doi.org/10.25728/pu.2018.5.1

МЕТОДЫ ПОСТРОЕНИЯ ОПТИМАЛЬНЫХ ОЧЕРЕДЕЙ ВОЗДУШНЫХ СУДОВ НА ПОСАДКУ[1]. Ч. 2. Методы приближенного решения

Г.С. Вересников, Н.А. Егоров, Е.Л. Кулида, В.Г. Лебедев

Рассмотрены методы приближенного решения статической задачи формирования оптимальной очереди воздушных судов на посадку, которые не гарантируют получение точного решения, но дают возможность получить приемлемое решение, удовлетворяющее предъявленным требованиям. Отмечено, что они, как правило, представляют собой синтез метаэвристического метода глобальной оптимизации для получения последовательности воздушных судов на посадку и локального точного метода для получения оптимального решения для полученных последовательностей. Представлен краткий обзор некоторых из них.

Ключевые слова: оптимальная очередь на посадку, целевая функция, генетические алгоритмы, глобальная и локальная оптимизация, меметические алгоритмы.

ВВЕДЕНИЕ

В первой части [1] обзора рассмотрены методы *точного* решения задачи формирования оптимальной очереди воздушных судов (ВС) на посадку (ФООП) с помощью методов линейного программирования и методов ветвей и границ.

Однако в настоящее время ведется большое число исследовательских работ, в результате которых предлагаются алгоритмы *приближенного* решения задачи ФООП, например, метод рассеивания и бионический метод, метод искусственных иммунных систем, табу-поиск, гибридные методы и др.

На ранних этапах для приближенного решения задачи ФООП предлагалось применять различные метаэвристические алгоритмы поисковой оптимизации. Основной недостаток алгоритмов данного класса заключается в их медленной сходимости к точному решению в окрестности глобального оптимума, так как они не дают возможности использовать локальную информацию о ландшафте исследуемой функции, что ограничивает их применение для решения задач ФООП, где время вычислений является критическим фактором.

Одно из перспективных современных направлений в области эволюционных вычислений состоит в применении меметических алгоритмов. Они относятся к популяционным метаэвристическим методам поисковой оптимизации, дополненным реализацией какого-либо метода локальной оптимизации, уточняющего решение в процессе поиска. Первоначально меметические алгоритмы были предложены как один из способов повышения эффективности генетических алгоритмов. Однако интеграция глобального и локального поисков в меметических алгоритмах применяется для построения поисковых алгоритмов также на основе других популяционных алгоритмов, например, алгоритмов муравьиных колоний и др.

Во второй части настоящего обзора рассматриваются генетические и меметические алгоритмы: алгоритм рассеянного поиска и бионический алгоритм, муравьиный алгоритм, алгоритм поиска и оценки последовательности приземлений.

1. ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ

Большое число работ посвящено применению генетических алгоритмов для решения задачи ФООП. Это обусловлено преимуществами, которые предоставляют генетические алгоритмы для решения прикладных задач [2]:

[1] Работа выполнена при частичной финансовой поддержке РФФИ (проект № 18-08-00822) и Программы I.30 Президиума РАН.

— отсутствие ограничений на вид используемых данных и целевых функций для решения оптимизационных задач; генетические алгоритмы достаточно универсальны и могут значительно упростить формализацию рассматриваемой задачи, когда для других методов сделать это затруднительно;

— высокая адаптивность генетических алгоритмов, позволяющая с наименьшими затратами встраивать дополнительные эвристики;

— возможность применения генетических алгоритмов в задачах с большим числом параметров, для которых нет возможности получения точного решения за приемлемое время; выполняется направленный поиск не точного (оптимального решения), а набора допустимых решений, из которых последовательно выбираются лучшие;

— возможность многокритериальной оптимизации, когда формируется Парето-множество решений;

— естественная возможность распараллеливания при формировании потомков, определяемая особенностями алгоритма и позволяющая значительно увеличить скорость получения результата на современных вычислительных средствах.

Отметим и недостатки генетических алгоритмов, которые оказывают непосредственное влияние на решение прикладных задач:

— отсутствие надежных критериев для завершения работы алгоритма;

— проблема сходимости и отсутствия гарантии нахождения глобального экстремума.

Подходы к решению задачи ФООП с помощью генетических алгоритмов отличаются:

— способом кодирования решений;

— способом построения целевой функции;

— учитываемыми ограничениями;

— выбором параметров и правил выполнения генетических операций;

— дополнительными эвристиками, позволяющими улучшать локальное решение.

Обычно решение представляется в виде вектора, называемого хромосомой, который определяет позицию в последовательности посадок или время посадки ВС и номер назначаемой взлетно-посадочной полосы (ВПП). Если в хромосоме содержится последовательность посадок ВС, то дополнительно требуется процедура назначения с учетом ограничений и целевой функции времени посадки для каждого ВС. Если хромосома содержит время посадки, то необходимо учитывать возможность нарушения ограничений при генерации начальной популяции, выполнении операций скрещивания и мутации.

Одно из первых подробных описаний применения генетических алгоритмов к решению задачи ФООП приводится в работе [3], в соответствии с которой каждое ВС получает символьный код (идентификатор) и расписание формируется в виде, представленном в табл. 1.

Подобный принцип кодирования расписания встречается и у других авторов. Отметим, что при таком подходе возникает проблема реализации операции скрещивания хромосом, обеспечивающей отсутствие дублирования двух символьных идентификаторов, по сути означающее повторную посадку одного и того же ВС.

Заметим, что проблема дублирования двух одинаковых символьных идентификаторов является классической проблемой, возникающей при решении генетическими алгоритмами NP-полной задачи коммивояжера, когда необходимо построить оптимальный маршрут посещения всех городов, зайдя в каждый из них один раз. В этой задаче классические операции скрещивания не подходят из-за отсутствия гарантии формирования допустимых потомков. Поэтому для решений этой задачи были разработаны специальные виды операций скрещивания: OX, CX, PMX и др. [4].

На экспериментальных данных, приведенных в работе [3], задача ФООП успешно решается для 40 ВС и двух ВПП, но при одинаковом интервале между посадками ВС (3 мин) и ограничениях на раннее прибытие, что значительно упрощает расчет целевой функции и учет ограничений в алгоритме.

В частности, в расчет целевой функции входят расчеты:

— значения $x_{ij} = x_{1j} + 3P_{ij}$, которые определяют время приземления для ВС i на ВПП j, где x_{1j} — время приземления первого ВС на ВПП j, P_{ij} — номер в очереди посадок ВС i на ВПП j;

— отклонения $d_{ij} = x_{ij} - T_{ij}$, где T_{ij} — оптимальное время приземления ВС i на ВПП j;

— целевой функции, определяющей уровень M_{ij} удовлетворенности временем посадки для каждого ВС:

$$M_{ij} = \begin{cases} d_{ij}^2, \text{ если } d_{ij} \geq 0, \\ \sqrt{|d_{ij}|}, \text{ если } 0 > d_{ij} \geq -3, \\ d_{ij}^4, \text{ если } d_{ij} < -3. \end{cases}$$

Таблица 1

Кодирование решения

Код ВС	CZB	TXK	RMD	HYK	...	NDU	TAH
Номер очереди	1	1	2	2	...	20	20
ВПП	1	2	1	2	...	1	2

Целевая функция рассчитывается как сумма индивидуальных функций всех ВС:

$$F = \sum_{i=0}^{R} \sum_{j=0}^{P_i} M_{ij},$$

где R — число ВПП, P_i — число ВС, приземляющихся на ВПП i.

В работе [5] рассматривается задача с большим числом ограничений, учитывается категория турбулентности ВС и используются тридцатисекундные слоты для посадки. Каждый ген хромосомы содержит время посадки для ВС и номер ВПП. Время посадки задается целым числом, определяющим число тридцатисекундных интервалов, что позволяет значительно снизить вычислительную сложность задачи. Время посадки первого ВС равно нулю.

Если в хромосоме кодируется время посадки на ВПП, становится возможным появление некорректных хромосом с нарушением ограничений разделения между различными ВС. Поэтому требуется введение дополнительного критерия, позволяющего уменьшать число некорректных хромосом в каждом следующем поколении. Один из подходов, характерных для генетических алгоритмов, состоит в создании системы штрафов. В работе [5] целевая функция вычисляется таким образом:

Fitness = INVALID_PENALTY(If Applicable) +
+ CLASH_PENALTY(If Applicable) +
+ No_planes_too_close * TOO_CLOSE_PENALTY +
+ No_planes_adj_too_close * ADJ_PENALTY +
+ total_early * EARLY_PENALTY + total_delay *
* DELAY_PENALTY

Таблица 2

Штрафы для целевой функции

Штраф	Описание	Значения
INVALID_ PENALTY	Некорректное расписание по любой причине	100
CLASH_ PENALTY	Когда два ВС заходят на посадку в одно и то же время	10
TOO_CLOSE_ PENALTY	ВС заходят на одну и ту же ВПП без соблюдения временного интервала	10
ADJ_ PENALTY	ВС заходят на соседнюю ВПП без соблюдения временного интервала	10
EARLY_ PENALTY	ВС заходят на посадку слишком рано	3
DELAY_ PENALTY	ВС заходят на посадку слишком поздно	1

Из табл. 2 видно, что предпочтение отдается соблюдению ограничений. Заход на посадку до планового времени менее предпочтителен, чем после него.

В работе [4] также решается динамическая задача, когда частично меняется состав ВС, входящих в расписание, на основе учета решений, сформированных в результате предыдущего запуска алгоритма. Это приводит к улучшению начальной целевой функции и увеличению валидных расписаний в популяции. По словам авторов, представленный алгоритм показал достаточную вычислительную производительность для его применения в режиме реального времени.

2. МЕМЕТИЧЕСКИЕ АЛГОРИТМЫ

2.1. Основные определения

Рассматривается задача в классической постановке [6]. Сначала рассмотрим представление и оценку особей в них.

Особь — это представление возможного решения задачи. Для задачи ФООП на несколько ВПП особь должна содержать информацию о запланированном времени посадки и номере ВПП для каждого ВС. Таким образом, особь содержит две переменные для каждого ВС: y_i — доля времени, которое проходит до запланированного времени посадки на интервале $[E_i, L_i]$, где E_i и L_i — самое раннее и самое позднее допустимые времена приземления i-го ВС, r_i — номер ВПП, $i = 1, ..., P$, где P — число ВС.

Переменная y_i определяется соотношениями:

$$x_i = E_i + y_i(L_i - E_i),$$

$$0 \le y_i \le 1 \quad \forall i = 1, ..., P.$$

Особь для *P* воздушных судов

ВС	1	2	...	P
y_i	y_1	y_2	...	y_P
ВПП	r_1	r_2	...	r_P

Нелинейная задача. Как уже отмечалось, ключевым фактором в управлении воздушным движением служит отклонение от целевого времени посадки.

Если отклонение $d_i = x_i - T_i$, $i = 1, ..., P$, положительное, то ВС идет на посадку после своего целевого времени. Это не идеальная ситуация, и для этого ВС устанавливается вклад в целевую функцию $-d_i^2$. Если отклонение отрицательное, то ВС идет на посадку прежде его целевого времени посадки, что предпочтительнее, поэтому значение

соответствующего вклада $+d_i^2$. Целевая функция представляет собой общий вклад всех ВС:

$$fitness = \sum_{i=1}^{P} D_i,$$

$$\text{где } D_i = \begin{cases} -d_i^2, \text{ если } d_i \geq 0, \\ +d_i^2, \text{ в противном случае.} \end{cases} \quad (1)$$

Необходимо максимизировать целевую функцию.

Преимущество такой постановки заключается в том, что, если задана последовательность посадки, можно вычислить оптимальные сроки посадки в полиномиально ограниченное время.

Линейная задача. Целевая функция для минимизации стоимости отклонения от целевого времени посадки, подробно рассмотренная в первой части [1] обзора, имеет вид:

$$fitness = \sum_{i=1}^{P} (g_i\alpha_i + h_i\beta_i), \quad (2)$$

где g_i — штраф за единицу времени приземления i-го ВС, $i = 1, ..., P$, раньше времени T_i; h_i — штраф за единицу времени приземления i-го ВС, $i = 1, ..., P$, позднее времени T_i; переменная α_i — насколько i-е ВС, $i = 1, ..., P$, приземляется раньше времени T_i; переменная β_i — насколько i-е ВС, $i = 1, ..., P$, приземляется позднее времени T_i.

В этом случае цель заключается в том, чтобы все ВС приземлились как можно ближе к своему целевому времени. Такая целевая функция может применяться при умеренной плотности трафика.

Оценка особи включает в себя значение целевой функции и значение непригодности.

Значение целевой функции определяет результативность особи. Если решается нелинейная задача (1), то целевая функция максимизируется, т. е. чем больше ее значение, тем лучше решение. Если решается линейная задача (2), то целевая функция минимизируется, т. е. в этом случае чем меньше ее значение, тем лучше решение.

Значение непригодности определяется по формуле

$$unfitness = \sum_{\substack{i=1 \\ j \neq i \ x_j \geq x_i \ r_i = r_j}}^{P} \sum_{j=1}^{P} \max(0, S_{ij} - (x_j - x_i)),$$

где S_{ij} — минимальный интервал между посадкой ВС j за ВС i, j, $i = 1, ..., P$.

Значение непригодности всегда будет равно нулю, если разделение выполнено, и строго положительно в противном случае. Чем выше значение непригодности, тем более неосуществимо решение.

Значение целевой функции (*fitness*) и значение непригодности (*unfitness*) используются для сравнения отдельных особей. Особь с нулевым значением непригодности лучше, чем любая особь с положительным значением непригодности. Из двух особей с нулевым значением непригодности (обе особи допустимы) лучше та, у которой лучше значение целевой функции.

1.1. Генетический алгоритм, включающий в себя процедуру локальной оптимизации

Работа [7] основывается на классической постановке задачи [6], подробно рассмотренной в первой части [1] настоящего обзора. В ней описывается алгоритм решения задачи ФООП на одну ВПП, учитываются ограничения разделения между ВС с учетом категории турбулентности для каждого ВС, ограничения на временные окна посадок ВС. Для проверки эффективности предлагаемых алгоритмов авторы воспользовались реальными данными, полученными в лондонском аэропорту Хитроу.

Задача решается для нелинейной целевой функции (1).

Одна из ключевых особенностей алгоритма состоит в формировании последующей популяции на основе значений целевой функции (*fitness*) и значений функции непригодности (*unfitness*).

Значения *fitness* и *unfitness* позволяют для каждого ребенка выполнить разделение членов текущей популяции на 4 группы, как показано на рис. 1.

Разделение на группы используется для выбора особей, которые будут заменены ребенком. Сначала из популяции удаляются особи из группы 1, у которых значения обеих функций хуже, чем у добавляемого потомка. Если таковых нет, то удаляются особи из группы 2, у которых значение *fitness*

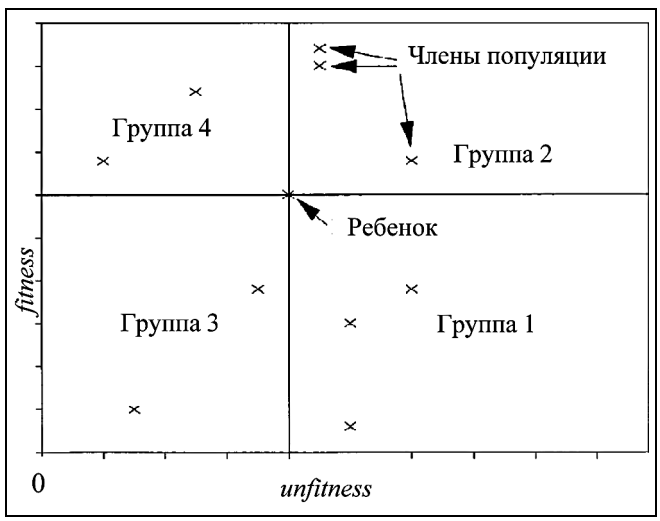

Рис. 1. Группы для замены популяции

лучше, а значение *unfitness* хуже, чем у добавляемого потомка и т. д. При таком способе отбора последовательно улучшаются средние значения функций *fitness* и *unfitness* в популяции.

В работе [7] приводится алгоритм, который по сути является меметическим, поскольку помимо генетического алгоритма поиска решений включает в себя процедуру локальной оптимизации полученных решений. Процедура локальной оптимизации не приводится, но далее в обзоре п. 2.4.6 приводится процедура локальной оптимизации, позволяющая в случае целевой функции вида (1) вычислить оптимальные сроки посадки в полиномиально ограниченное время.

В работе [7] представлены результаты, полученные на фактических эксплуатационных данных, относящихся к посадкам ВС в аэропорту Хитроу. Эти данные показывают, что алгоритм мог бы улучшить решения по управлению воздушным движением на 2—5 % в смысле сокращения промежутка времени, необходимого для посадки всех рассматриваемых ВС.

2.3. Метод рассеянного поиска и бионический алгоритм [7]

2.3.1. Общая схема метода рассеянного поиска и бионического алгоритма

В работе [8] рассматриваются две задачи в классической постановке [6], различающиеся своими целевыми функциями (1) и (2), и приводятся два алгоритма их решения.

Особенности метода *рассеянного поиска* (Scatter Search): особи не ограничиваются двоичным представлением; родителей может быть больше двух; скрещивание родителей строится как их линейная комбинация; локальная процедура улучшения применяется ко всем особям.

Общая схема метода рассеянного поиска

1. Генерировать начальную популяцию, называемую эталонным набором.
2. Улучшить каждую особь в эталонном наборе.
3. *Повторять:*

— выбрать подмножество из эталонного набора;

— создать новую особь как линейную комбинацию этого подмножества;

— улучшить новую особь;

— обновить эталонный набор

до тех пор, пока не выполнены условия окончания алгоритма.

Выбрать лучшее из получившихся решений.

Особенности *бионического алгоритма* (Bionomic Algorithm): этап созревания для улучшения особей; структурное построение родительских наборов, основанное на графе, который отражает структуру популяции; родительский отбор на основе целевой функции и расстояния между особями; поколенческий подход для замены популяции.

Общая схема бионического алгоритма

1. Генерировать начальную популяцию.
2. Улучшить каждую особь в эталонном наборе.
3. *Повторять*:

— построить граф, представляющий структуру популяции;

— выбрать родительские подмножества из этого графа;

— создать новую особь как линейную комбинацию для каждого такого родительского подмножества;

— улучшить каждую новую особь;

— обновить популяцию с некоторыми из лучших новых особей

до тех пор, пока не выполнены условия окончания алгоритма.

Выбрать лучшее из получившихся решений.

Основные шаги алгоритмов:

— генерация начальной популяции;

— выбор родителей;

— генерация особи;

— тест на дублирование;

— локальное улучшение особи;

— обновление популяции.

Многие из этих шагов общие для обеих эвристик либо потому, что они соответствуют основным функциям популяционных эвристик, либо потому, что они непосредственно относятся к задаче ФООП. Основным шагом, который отличается — выбор родителей.

2.3.2. Генерация начальной популяции

Большинство особей генерируются случайным образом. Чтобы создать случайные особи, случайным образом генерируются два значения для каждого ВС: случайное дробное значение от 0 до 1 и номер ВПП — случайное целое от 1 до R.

Три особи могут быть получены при использовании эвристики в виде трех последовательностей ВС, которые можно определить с помощью раннего, целевого или позднего времени посадки. Как и со случайными особями, не гарантируется, что особи являются допустимыми. Численность популяции составляет 100 особей, поэтому начальная популяция включает в себя три особи, созданные при помощи эвристики, а остальные сгенерированы случайным образом.

2.3.3. Выбор родителей

Выбор родителей в алгоритме рассеянного поиска — турнирный отбор на основе индивидуальной приспособленности. Из двух особей, выбранных случайным образом из популяции, в качестве родителя выбирается особь с лучшим значением целевой функции. Схема турнирного отбора повторяется три раза, чтобы выбрать три особи. По окон-

чании этапа отбора родителей образуется один родительский набор, содержащий три особи.

Выбор родителей в бионическом алгоритме — это структурированная процедура, основной принцип которой — дать больше возможностей для особей с лучшим значением целевой функции, чтобы при этом особи, слишком близкие друг к другу, не могли быть выбраны как родители одновременно.

Структура популяции представляется в виде графа смежности. Каждая особь текущей популяции представляется в виде вершины. Вершинам приписана частота включения F_n, которая соответствует номеру особи в последовательности особей, отсортированной по значению целевой функции. Вершина с более высокой частотой включения может появляться более часто в родительском наборе. Для нелинейной задачи особи сортируются по возрастанию значения целевой функции. Для линейной задачи особи сортируются по убыванию значения целевой функции.

Вводятся обозначения:

N — размер популяции и также число вершин в графе смежности;

F_n — ранг особи n и частота включения вершины n, $n = 1, ..., N$; $F_n \in \{1, ..., N\}$.

Для количественной оценки близости двух особей друг к другу вводится понятие расстояния между ними. Значение расстояния используется для того, чтобы определить, существует ли ребро между двумя соответствующими вершинами в графе смежности.

Определяются:

d_{ij} — общая оценка расстояния между особями i и j;

d_{ij}^k — оценка расстояния, соответствующая ВС с номером k между особями i и j;

y_i^k — y_i для ВС с номером k в особи i;

r_i^k — номер ВПП, связанный с ВС с номером k в особи i.

Определяется

$$d_{ij} = \sum_{k=1}^{P} d_{ij}^k, \quad d_{ij}^k = \begin{cases} 1, \text{если } r_i^k \neq r_j^k, \\ \left| y_i^k - y_j^k \right|, \text{в противном случае,} \end{cases}$$

$$\forall (i, j) \in \{1, ..., N\}^2;\ i \neq j.$$

Помимо оценки d_{ij}, определяется порог Θ. Расстояния для всех пар особей сравниваются с этим порогом. Если $d_{ij} < \Theta$, особи i и j считаются близкими друг к другу, и должно быть ребро между соответствующими вершинами в графе смежности. В реализации порог Θ первоначально устанавливался равным $P/10$. Если граф смежности, построенный с этим значением, содержит более чем половину максимального числа ребер неориентированного графа с N вершинами ($N(N-1)/2$ ребер), то порог Θ уменьшается в два раза и рассчитывается новый граф. Уменьшение порога Θ было введено, поскольку предварительные результаты расчетов показали, что чем больше ребер в графе смежности, тем менее разнообразны автоматически получаемые из графа смежности наборы родителей.

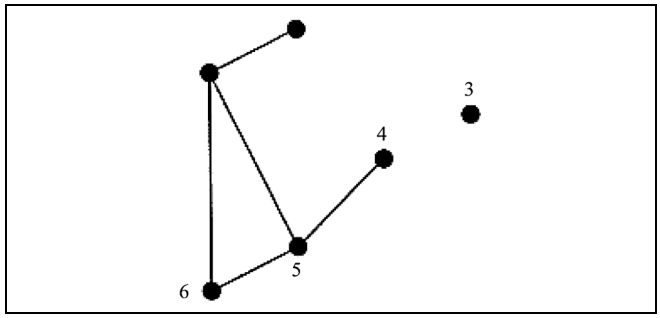

Рис. 2. Пример графа для иллюстрации максимальных независимых множеств

В бионическом алгоритме наборы родителей задаются путем вычисления максимального независимого множества графа смежности — набора максимального числа элементов, где ни одна пара из выбранных вершин не соединена ребром.

Концепция возможных максимальных независимых множеств для графа смежности из шести вершин проиллюстрирована на рис. 2. Соответствующие этому графу максимальные независимые множества, {2, 3, 4, 6}, {2, 3, 5} и {1, 3, 4}. Для создания родительского набора можно взять любое из максимальных независимых множеств.

Процесс вычисления максимального независимого набора

Повторять:

— выбрать одну вершину из узлов в случайном порядке;

вставить эту вершину в родительский набор;

удалить выбранную вершину и все связанные с ней ребрами вершины из графа

до тех пор, пока множество вершин графа не пусто.

Как только этот процесс выполнен, новый родительский набор P_s построен. При этом частота включения вершин, присутствующих в наборе P_s, уменьшается на единицу (т. е. $F_n = F_n - 1 \forall n \in P_s$). Если значение частоты включения вершины становится равным нулю, вершина удаляется из главного графа смежности. Таким образом, особь n будет выбрана в родительский набор F_n раз.

Вычислительная сложность, связанная с построением графа смежности $O(PN^2)$ и сложности, связанные с генерацией всех родительских набо-

ров из графа смежности (в худшем случае) $O(N^3)$. Следовательно, общая в худшем случае вычислительная сложность здесь составляет $O(PN^2 + N^3)$. В работе [8] утверждается, что алгоритмы тестировались и хорошо работают при фиксированной численности популяции $N = 100$. При фиксированном N связь между временем вычислений и размером задачи P линейная.

2.3.4. Генерация особи

Для обоих алгоритмов процедура генерации особи производит одну особь от каждого из наборов родителей, и эта особь может быть допустимой или недопустимой.

Процедура генерации особи

Пусть $p(1), p(2), ..., p(K)$ — родители в родительском наборе.

Генерируется K случайных чисел $0 \le w_i \le 1$ с номерами $i = 1, ..., K$,

Положим $W = w_1 + ... + w_K$ и $w_i = w_i/W$ для $i = 1, ..., K$

$$y^k_{child} = \sum_{i=1}^{K} w_i y^k_{p(i)} \quad \forall k = 1, ..., P.$$

Повторять последовательно для каждого ВС:

— выбрать одного из k-родителей случайным образом;

— назначить номер его ВПП в качестве номера ВПП воздушного судна у генерируемой особи

до тех пор, пока не все ВС рассмотрены.

Метод рассеянного поиска производит только одну особь в каждом поколении, описанная выше процедура выполняется один раз, используется набор из трех выбранных родителей. В отличие от этого, бионический алгоритм генерирует несколько родительских наборов, и описанная выше процедура выполняется один раз для каждого из них, производя одну особь для каждого набора родителей.

2.3.5. Тест на дублирование

Для поддержания разнообразия в популяции новые особи отбрасываются, если они дублируют особь, уже присутствующую в популяции. Новая особь проверяется на дублирование среди всех существующих особей сразу после ее создания.

В случае одной ВПП тест на дублирование проверяет, являются ли последовательности посадки двух особей идентичными, т. е. все ВС приземляются в том же порядке. Если так, то обе особи представляют одно и то же решение. Отметим здесь, что локальные процедуры улучшения (описанные далее) таковы, что если последовательности посадки совпадают, то решения после локального улучшения также будут одинаковыми. Понятно, что, в смысле вычислительной сложности, проверка на дублирование должна проводиться перед началом локального улучшения. В случае нескольких ВПП проверка на дублирование несколько сложнее.

2.3.6. Локальное улучшение особи

Процедура улучшения особи зависит от вида целевой функции и применяется к фиксированной последовательности посадок. В результате определяются оптимальные времена посадок для этой последовательности.

Для нелинейной задачи можно вычислить оптимальные сроки посадки с помощью простой процедуры полиномиальной сложности. Она применяется независимо к каждой последовательности посадки и заключается в следующем.

Пусть $q(s)$ — s-е ВС в упорядоченной последовательности посадок на ВПП.

Присвоить $s = 1$ и $x_{q(1)} = E_{q(1)}$.

Повторять:

— присвоить $s = s + 1$;

— присвоить $x_{q(s)} = \min[L_{q(s)}, \max[E_{q(s)}, x_{q(t)} + S_{q(t),q(s)}\ t = 1, ..., s - 1]]$

до тех пор, пока все ВС в упорядоченной последовательности не будут рассмотрены.

Если в результате этой процедуры вычисляются времена посадок, которые осуществимы (удовлетворяют временным ограничениям разделения), то они оптимальны для данной последовательности посадок. Вычислительная сложность данной процедуры $O(P^2)$.

Для линейной целевой функции процедура улучшения соответствует решению задачи линейного программирования (ЗЛП), не содержащей целочисленных переменных. Эта процедура применяется для фиксированной последовательности посадок, поэтому (в смысле математической постановки задачи) все двоичные переменные известны. Переменные, которые нужно вычислить, — времена посадок для каждого ВС. Эта задача решается с помощью пакета программ оптимизации CPLEX.

Однако, с учетом приземления на каждую ВПП, полученные решения ЗЛП могут быть реализуемы или нет. Если это возможно, CPLEX вычисляет оптимальные времена посадки для заданной последовательности посадки на основе первоначальной целевой функции. Если решение ЗЛП для какой-либо конкретной ВПП нереализуемо, новая ЗЛП моделируется на основе той же последовательности посадок для минимизации значения непригодности. Если предположить (для простоты обсуждения), что решение ЗЛП для всех ВПП нереализуемо, ставится новая ЗЛП:

$$\min \sum_{i=1}^{P} \sum_{\substack{j=1 \\ j \ne i\ x_j \ge x_i\ r_i = r_j}}^{P} \max[(0, S_{ij} - (x_j - x_i)).$$

2.3.7. Обновление популяции

После генерации новой особи (не дубликатной) и ее улучшения, она включается в популяцию. Нужно сохранять численность популяции постоянной и, следовательно, одну особь нужно выбрать для удаления. В алгоритме рассеянного поиска удаляется особь с худшим значением целевой функции.

Исследования показали, что сохранение многообразия популяции необходимо для эффективной работы алгоритма. Поэтому включение новых особей с худшим значением целевой функции, чем у остальных представителей популяции, оправданно.

Схема обновления популяции для бионического алгоритма, обычно представляемая в литературе, соответствует поколенческому подходу, когда новые особи-дети вставляются в популяцию на той же стадии. Однако предварительные вычислительные опыты показывают, что следование такому подходу для задачи ФООП не приводит к хорошим результатам. В частности, добавление многих особей-детей в каждом поколении быстро приводит к однородности популяции и получаются результаты низкого качества.

Поэтому в популяцию добавляется лучшая из сформированных особей (с наименьшим значением непригодности или, при одинаковом значении непригодности, с лучшим значением целевой функции).

В работе [8] представлены результаты расчетов для 13 примеров из OR-Library [9], с участием от 10 до 500 ВС, с разным числом ВПП и двумя целевыми функциями.

Для нелинейной задачи лучшие результаты получены благодаря применению бионического алгоритма, а для линейной задачи лучшие результаты дает алгоритм рассеянного поиска.

Что касается времени вычислений, то бионический алгоритм требует больших временных затрат, тем не менее, оба представленных алгоритма требуют вполне разумных временных затрат относительно размерности решаемой задачи и, по утверждению авторов, превосходят все ранее представленные в литературе алгоритмы.

2.4. Табу-поиск

В работе [10] в качестве дополнительной локальной процедуры оптимизации, интегрированной в генетический алгоритм, применяется табу-поиск. Идея табу-поиска заключается в том, что на лучшие решения налагается временный запрет на их использование в алгоритме в процессе формирования новых решений. Это позволяет в процессе поиска решения избегать локальных минимумов и искать решения, более близкие к глобальному оптимуму. Авторы статьи, в качестве доказательства эффективности алгоритма с табу-поиском, приводят результаты сравнения с различными алгоритмами, в частности, с алгоритмом рассеянного поиска. Отмечается, что в большинстве случаев с помощью табу-поиска находятся более близкие к оптимуму решения и обеспечивается лучшая вычислительная эффективность алгоритма.

2.5. Муравьиные алгоритмы [11]

Широкое применение для решения задачи ФООП получили методы, основанные на муравьиных алгоритмах. Муравьиный алгоритм — это метаэвристический алгоритм оптимизации, основанный на подражании поведению муравьиной колонии. Он является одним из эффективных полиномиальных алгоритмов, которые успешно применяются для поиска приближенных решений задачи построения оптимальных маршрутов на графах. Этот алгоритм по эффективности нередко сравнивается с генетическими алгоритмами [12, 13], он зачастую выигрывает по вычислительной производительности и качеству найденного решения в задачах большой размерности.

Предложенный в работе [11] алгоритм решения задачи ФООП, хорошо отражает основные принципы применения муравьиных алгоритмов к подобному типу задач.

Принцип выбора пути (последовательности ВС) для каждого «муравья» авторы иллюстрируют графом, представленном на рис. 3. Сначала выбирается ВПП, затем выбираются ВС на посадку. Когда вся последовательность ВС на посадку сформирована, но не достигнут критерий остановки работы алгоритма, начинается следующая итерация алгоритма, и данная процедура повторяется.

Для выбора k-м муравьем ВПП для очередного ВС используется вероятностный критерий, который рассчитывается по формуле

$$P_{Dr}^{k} = \begin{cases} \arg\min\limits_{r = 1, \ldots, R} \left(\arg\min\limits_{j \in Candidate_k} (x_{i_0}^{r} + S_{i_0 j}) \right), \text{если } q < q_0 \\ r_0 \text{ в противном случае,} \end{cases}$$

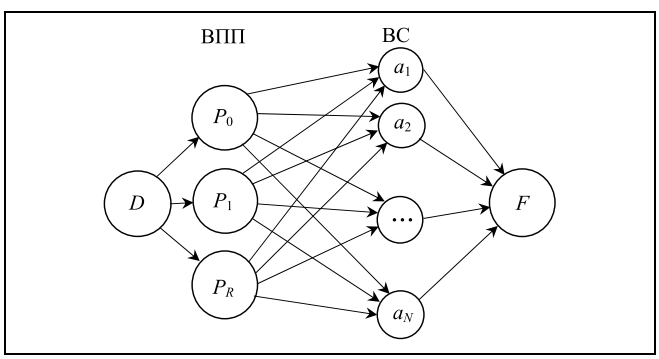

Рис. 3. Графическое представление выбора пути для муравьиной колонии

где q — случайное число, которое генерируется в диапазоне от 0 до 1; q_0 — константа в диапазоне от 0 до 1, параметр алгоритма; r_0 — индекс, выбираемый случайно в диапазоне от 0 до R; $Candidate_k$ — список ВС, которые к текущему моменту еще не выбраны муравьем k; $x^r_{i_0}$ — время посадки последнего ВС на полосу r; $S_{i_0 j}$ — безопасный интервал посадки ВС j за ВС i_0.

В соответствии с данной эвристикой, если $q_0 = 1$, то наибольший приоритет всегда будет иметь ВПП, освобождающаяся в ближайшее время. В то же время константа $q_0 < 1$ позволяет с некоторой вероятностью переключиться на случайный выбор полосы, что обеспечивает диверсификацию генерируемых решений.

Для выбора ВС также используется предлагаемая авторами эвристика. Рассчитывается вероятностный критерий $p^k_{rj}(t)$, который является взвешенной величиной, учитывающей текущее видение «муравья» и предыдущий опыт муравьиной колонии.

Текущее видение «муравья» определяется величиной

$$\eta_{rj} = \left(\frac{1}{Priority(j)+1}\right)^{\beta_1}\left(\frac{1}{Cost\_penalty(j)+1}\right)^{\beta_2},$$

где $Priority(j)$ — определяет приоритет для j-го ВС; $Cost\_penalty(j)$ — штраф для j-го ВС, β_1 и β_2 — коэффициенты, позволяющие регулировать соотношение между приоритетом и штрафом.

В соответствии с основными принципами муравьиных алгоритмов создается и инициализируется нулями матрица для хранения значений накопленных феромонов τ_{ij}, соответствующих посадке ВС j после ВС i. Предыдущий опыт муравьиной колонии, оставляемый феромонами «муравья», учитывается на каждой итерации алгоритма в матрице τ_{ij} по формуле:

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \Delta\tau_{ij}(t),$$

где ρ — коэффициент испарения феромонов; $\Delta\tau_{ij}(t)$ — величина, добавляемая в общую сумму накопленных феромонов. Она вычисляется по формуле

$$\Delta\tau_{ij}(t) =$$

$$= \begin{cases} \dfrac{Q}{C}, \text{ если } (i,j) \text{ принадлежит лучшему решению,} \\ 0, \text{ иначе,} \end{cases}$$

где Q — константа, параметр алгоритма; C — штраф, соответствующий лучшему решению в итерации t.

Вероятностный критерий $p^k_{rj}(t)$ рассчитывается по формуле

$$p^k_{rj}(t) = \begin{cases} \dfrac{(\tau_{rj})^{\alpha}(\eta_{rj})^{\beta}}{\sum\limits_l (\tau_{rl})^{\alpha}(\eta_{rl})^{\beta}}, \text{ если } j \in Candidate_k, \\ 0, \text{ если } j \notin Candidate_k, \end{cases}$$

где α и β — параметры, позволяющие при выборе ВС регулировать приоритет между текущим видением «муравья» и предыдущим опытом муравьиной колонии.

В работе [11] рассматриваются два алгоритма, один из которых называется улучшенной версией и позволяет находить наиболее близкие к оптимальному решения, но с более значительными вычислительными затратами.

Предложенные авторами алгоритмы, протестированы для посадки от 10 до 50 ВС на три ВПП. При преимуществе улучшенной версии алгоритма в близости найденного решения к оптимальному время работы одного алгоритма не превышает 10 с, а улучшенного 168 с. В следующих работах, в которых приведены дополнительные результаты исследования, улучшенная версия алгоритма получила название меметический алгоритм [14]. Данное название указывает на то, что используется дополнительная локальная процедура поиска путей улучшения решения, не являющаяся элементом муравьиного алгоритма. Отметим, что многие авторы, по сути, применяют меметические алгоритмы, не употребляя в тексте статьи данный термин. Например, в работе [15] применяется интеграция муравьиного алгоритма и метода линейного программирования, что позволяет улучшить локальные решения и повысить общую вычислительную эффективность решения рассматриваемой задачи. В частности, метод линейного программирование применяется как метод локальной оптимизации, чтобы обеспечить минимальные допустимые интервалы посадки между ВС при фиксированной последовательности посадок.

В результате проведенных экспериментов на тестовых данных авторы делают вывод, что предложенный муравьиный алгоритм во многих случаях имеет более высокую вычислительную производительность по сравнению с другими методами [11].

2.6. Метод поиска и оценки последовательности приземлений воздушных судов

Один из универсальных методов поиска и оценки последовательности приземления представлен в статье [16]. В отличие от других работ, основное внимание уделено разработке общей структуры алгоритма, которая не подпадает под ограничения конкретной модели и может быть легко адаптирована к различным задачам ФООП. Такой подход не гарантирует улучшения работы по сравнению с существующими алгоритмами, но он в состоянии хорошо обобщать различные задачи ФООП и генерировать приемлемые результаты как по точности, так и по эффективности. Поскольку все различные задачи ФООП могут быть сформулированы как задачи оптимизации на основе подмножества перестановок [5, 17], предлагается расширяемая структура для решения задач ФООП на основе перестановок. Разработан ряд методов, в частности, метод SSE (Sequence Searching and Evaluation), в котором процесс решения задачи заключается в последовательном выполнении двух этапов:

— сведение задачи ФООП к задаче определения перестановки и поиска новых оптимальных перестановок в соответствии с определенными правилами;

— поиск решения для каждой такой перестановки, т. е. определение оптимального времени посадки для каждого ВС, оценка качества решений и фиксация лучших на основе исходной целевой функции и ограничений.

В результате итеративного выполнения этих этапов получается окончательное оптимальное решение задачи.

В качестве модуля поиска перестановки предлагается воспользоваться алгоритмом оценки распределения (Estimation of Distribution Algorithm, EDA). Отмечается, что EDA — алгоритм стохастической оптимизации, который приобрел широкую популярность в области эволюционных вычислений [18]. Алгоритм строит вероятностную модель, выражающую взаимосвязи между решениями в EDA. Путем выборки из этой вероятностной модели получается перестановка нового поколения. Алгоритм выполняет поиск оптимального решения до тех пор, пока не будет достигнута заданная точка останова. Этот алгоритм предлагает общую схему оптимизации сложной системы и не нуждается в какой-либо информации, кроме целевой функции. Учитывая эти особенности, EDA вполне соответствует поставленным требованиям. В последние годы были предприняты многочисленные усилия для решения задач на основе перестановок с помощью EDA, таких как алгоритм одномерного граничного распределения (Univariate Marginal Distribution Algorithm, UMDA), алгоритм байесовских сетей (Estimation of Bayesian Networks Algorithm, EBNAs) и др.

В статье [16] используются модель Мэллоуса и расстояние Kendall-τ в рамках EDA, поскольку они применимы для решения задачи ФООП на основе перестановок [19, 20]. Модель Мэллоуса представляет собой модель экспоненциальной вероятности на основе расстояния над перестановочными пространствами. Учитывая расстояние d над перестановками, модель Мэллоуса может быть определена центральной перестановкой σ_0 и параметром протяженности θ [21]. В частности, явный вид распределения вероятностей по пространству перестановок может быть представлен как:

$$P(\sigma) = \frac{1}{\psi(\theta)} e^{-\theta d(\sigma, \sigma_0)},$$

где σ — означает перестановку, а $\psi(\theta)$ — константа нормировки:

$$\psi(\theta) = \prod_{j=1}^{n-1} \frac{1 - e^{-(n-j+1)\theta}}{1 - e^{-\theta}},$$

n — число перестановок в множестве $\{\sigma_1, ..., \sigma_n\}$.

Расстояние Kendall-τ наиболее часто используется в модели Мэллоуса [21]. Если есть две перестановки $\sigma_1 = [\sigma_1(1), \sigma_1(2), ..., \sigma_1(n)]$ и $\sigma_2 = [\sigma_2(1), \sigma_2(2), ..., \sigma_2(n)]$, то расстояние Kendall-τ подсчитывает общее число попарных несоответствий между ними, т. е. минимальное число смежных перестановок, требуемых для преобразования σ_1 в σ_2, а именно:

$$d(\sigma_1, \sigma_2) = \sum_{i<j} I\{(\sigma_1(i) - \sigma_1(j))(\sigma_2(i) - \sigma_2(j)) < 0\},$$

где

$$I(x) = \begin{cases} 1, \text{если } x = \text{true}, \\ 0, \text{если } x = \text{false}. \end{cases}$$

Первоначальная популяция порождается путем генерации определенного числа случайных перестановок. Каждая перестановка представляет собой возможную (но не обязательно выполнимую) последовательность посадки ВС. На шаге обучения EDA модель Мэллоуса получается из популяции, т. е. множества перестановок, а затем новые перестановки генерируются путем выборки из вероятностной модели. В частности, метод SSE предполагает выполнение следующих шагов.

1. Центральная перестановка σ_0 изначально получается по всем индивидам в популяции. Задача определения оценки максимального правдоподобия центральной перестановки NP-сложная. Существует несколько способов решения этой задачи. Предлагается такой алгоритм расчета σ_0: сначала вычисляется среднее значение в каждой позиции,

затем первое положение σ_0, т. е. $\hat{\sigma}_0(1)$, присваивается позиции с наименьшим средним значением, далее положение $\hat{\sigma}_0(2)$ присваивается второй самой низкой позиции. Центральная перестановка будет окончательно получена, когда все значения будут назначены.

2. После получения центральной перестановки σ_0 новые перестановки генерируются путем выборки распределения вероятности P, определяемого центральной перестановкой σ_0 и заранее заданного параметра протяженности θ. После определения расстояния d по вероятностной модели алгоритм должен генерировать перестановку, которая имеет расстояние d от центральной перестановки.

3. В заключение оценивается осуществимость и качество как исходных, так и вновь сгенерированных перестановок, а популяция обновляется путем выбора усечения.

Итерации шагов 1—3 заканчиваются при достижении заданного критерия остановки.

Протяженность θ служит ключевым параметром контроля эффективности. Подходящее значение θ определяется экспериментально.

Далее авторы проводят анализ модуля оценки, который преобразует перестановки в допустимую форму решения в зависимости от сценария, описанного в задаче ФООП, а затем оценивает качество перестановок. Но в некоторых задачах не все перестановки могут быть преобразованы в допустимое решение. Например, некоторые ограничения, такие как ограничения на ранние/поздние времена посадки и ограничения на порядок движения ВС, делают некоторые перестановки недопустимыми. Если решение не удовлетворяет ограничениям, то оценка пригодности последовательности задается как бесконечно малая величина. Неосуществимые решения будут устранены во время следующих итераций.

Рассмотрим кратко некоторые приемы улучшения решения задачи ФООП, предлагаемые в работе [16].

Инициализация. Предположим, что необходимо инициализировать популяцию из N ВС и эти ВС могут быть сначала разделены на несколько групп в соответствии с их плановыми временами посадки. Например, ВС, которые прибывают в течение 10 мин, могут рассматриваться как группа, так что формируется ряд групп, в которых число ВС может несколько отличаться. Тогда в каждой группе произвольно порождаются перестановки. Полные перестановки ВС могут быть получены путем комбинирования перестановок разных групп. Эта простая операция ограничивает позиции ВС в определенном диапазоне на основе случайно сгенерированных перестановок. Несколько последовательностей, полученных по правилу «первым прибыл — первым обслужен», также добавляются к исходной популяции.

Локальный модуль поиска. Поскольку в процессе поиска EDA существует определенная случайность, необходим локальный модуль поиска, чтобы повысить эффективность. Предлагается достаточно простой, но эффективный способ получить лучшее решение — «скользящие и локальные перестановки» [16]. Скользящее окно, длиной k, используется для разбиения последовательности из N ВС на несколько подпоследовательностей. Процесс может занять много времени, поскольку потребуется $(N - k + 1)k!$ вычислений функции оценки перестановок. Поэтому данный модуль не следует применять слишком часто в течение процесса решения.

Общая схема алгоритма

1. Генерация популяции. Согласно модулю инициализации, случайным образом генерируется M различных перестановок для инициализации популяции, где M — численность популяции.

2. Инициализация вероятностной модели. Инициализация вероятностной модели P на основе модели Мэллоуса и расстояния Kendall-τ.

3. Поиск перестановки. Определяется центральная перестановка σ_0 текущей популяции. Согласно распределению вероятности P случайным образом генерируются M новых перестановок путем согласования соответствующего расстояния d.

4. Оценка перестановки. Сортируются $2M$ перестановок в соответствии с их оценкой. В популяции сохраняются M перестановок с лучшими показателями.

5. Локальный поиск. Чтобы получить лучшие решения, через определенное число итераций к популяции применяется модуль локального поиска.

6. Если итеративный процесс закончился, выводится лучшее решение, иначе переход к шагу 3.

Таким образом, SSE является меметическим алгоритмом, который предусматривает глобальный и локальный поиски [22].

В заключительной части работы [16] авторы приводят результаты моделирования предлагаемого алгоритма и сравнивают алгоритм с подходом Faye [23], основанным на приближении матрицы ограничений разделения и на дискретизации времени. Исходя из сравнения с другими методами решения задач ФООП, авторы делают вывод, что SSE обеспечивает общий метод решения различных видов задач ФООП за приемлемое время.

ЗАКЛЮЧЕНИЕ

Анализ постановок и методов решения задачи формирования оптимальной очереди воздушных судов на посадку показал, что эта область иссле-

дования представляет собой целый комплекс задач, сложность которых повышается при увеличении числа ограничений и внешних факторов, связанных с особенностями организации воздушного движения.

В первой части обзора представлены методы точного решения задачи формирования оптимальной очереди с помощью методов линейного программирования и методов ветвей и границ. Однако эти методы не всегда удовлетворяют требованиям решения в режиме реального времени, работы с большим потоком воздушного движения, безопасности и технической реализуемости, учета большого числа ограничений и внешних факторов.

Во второй части обзора рассмотрены методы приближенного решения задачи, которые не гарантируют получение точного решения, но дают возможность получить приемлемое решение, удовлетворяющее предъявленным требованиям. Как правило, они представляют собой синтез метаэвристического метода глобальной оптимизации для получения последовательности ВС на посадку и локального точного метода для получения оптимального решения для полученных последовательностей.

При оценке эффективности и выборе методов глобальной и локальной оптимизации для системы AMAN необходимо учитывать особенности конкретного аэропорта или аэроузла, интенсивность воздушного движения, особенности системы его организации.

ЛИТЕРАТУРА

1. *Вересников Г.С.*, *Егоров Н.А.*, *Кулида Е.Л.*, *Лебедев В.Г.* Методы построения оптимальных очередей воздушных судов на посадку. Ч. 1. Методы точного решения.
2. *Гладков Л.А.*, *Курейчик В.В.*, *Курейчик В.М.* Генетические алгоритмы: 2-е изд. — М.: Физматлит, 2006. — 320 с.
3. *Stevens G.* An approach to scheduling aircraft landing times using genetic algorithms. — RMIT: Department of Computer Science, 1995. — 41 p. — URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.50.9231 (дата обращения: 27.06.2018).
4. *Kumar N.*, *Karambir*, *and Kumar R.* A comparative analysis of PMX, CX and OX Crossover operators for solving traveling salesman problem // International Journal of Latest Research in Science and Technology. — 2012. — Vol. 1, N 2. — P. 98—101.
5. *Ciesielski V.*, *Scerri P.* Real Time Genetic Scheduling of Aircraft Landing Times // Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence. — 1998. — P. 360—364.
6. *Beasley J.E.*, *Krishnamoorthy M.*, *Sharaiha Y.M.*, *and Abramson D.* Scheduling aircraft landings — the static case // Transportation Science. — 2000. — Vol. 34, N 2. — P. 180—197.
7. *Beasley J.E.*, *Sonander J.*, *and Havelock P.* Scheduling Aircraft Landings at London Heathrow Using a Population Heuristic // Journal of the Operational Research Society. — 2001. — Vol. 52, N 5. — P. 483—493.
8. *Beasley J.E.*, *Pinol H.* Scatter search and bionomic algorithms for the aircraft landing problem // European Journal of Operational Research. — 2006. — Vol. 127, N 2. — P. 439—462.
9. *Beasley J.E.* OR-Library: distributing test problems by electronic mail // Journal of the Operational Research Society. — 1990. — Vol. 41, N 11. — P. 1069—1072.
10. *Bencheikh G.*, *Khoukhi F.* Hybrid algorithms for the multiple runway aircraft landing problem // International Journal of Computer Science and Applications. — 2013. — Vol. 10, N 2. — P. 53—71.
11. *Bencheikh G.*, *Boukachour J.*, *and Alaoui A.H.* Improved Ant Colony Algorithm to Solve the Aircraft Landing Problem // International Journal of Computer Theory and Engineering. — 2011. — Vol. 3, N 2. — P. 224—233.
12. *Частикова В.А.*, *Власов К.А.* Разработка и сравнительный анализ эвристических алгоритмов для поиска наименьшего гамильтонова цикла в полном графе // Фундаментальные исследования. — 2013. — Т. 10-1. — С. 63—66.
13. *Семенкина О.Е.*, *Семенкин Е.С.* О сравнении эффективности муравьиного и генетического алгоритмов при решении задач комбинаторной оптимизации // Актуальные проблемы авиации и космонавтики. — 2011. — Т. 1, № 7. — С. 338—339.
14. *Bencheikh G.*, *Boukachour J.*, *and Alaoui A.H.* A memetic algorithm to solve the dynamic multiple runway aircraft landing problem // Journal of King Saud University — Computer and Information Sciences. — 2016. — Vol. 28, N 1. — P. 98—109.
15. *Bo Xu.* An efficient Ant Colony algorithm based on wake-vortex modeling method for aircraft scheduling problem // Journal of Computational and Applied Mathematics. — 2017. — Vol. 317. — P. 157—170.
16. *Xiao-Peng Ji*, *Xian-Bin Cao*, *and Ke Tang.* Sequence searching and evaluation: a unified approach for aircraft arrival sequencing and scheduling problems // Memetic Computing. — 2016. — Vol. 8, N 2. — P. 109—123.
17. *Hu X.*, *Paolo E.* A ripple-spreading genetic algorithm for the aircraft sequencing problem // Evolutionary Computation. — 2011. — Vol. 19, N 1. — P. 77—106.
18. *Larracaga P.*, *Lozano J.A.* Estimation of distribution algorithms: A new tool for evolutionary computation. — Springer Science & Business Media, 2001.
19. *Ceberio J.*, *Irurozki E.*, *Mendiburu A.*, et al. A distance-based ranking model estimation of distribution algorithm for the flowshop scheduling problem // IEEE Trans. on Evolutionary Computation. — 2014. — Vol. 18, N 2. — P. 286—300.
20. *Ceberio J.*, *Irurozki E.*, *Mendiburu A.*, *and Lozano J.A.* A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems // Progress in Artificial Intelligence. — 2012. — Vol. 1, N 1. — P. 103—117.
21. *Ceberio J.*, *Mendiburu A.*, *and Lozano J.A.* Introducing the mallows model on estimation of distribution algorithms // Intern. Conf. on Neural Information Processing. Springer, Berlin, Heidelberg. — 2011. — P. 461—470.
22. *Moscato P.* On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. C3P Report, 826, California Institute of Technology, Pasadena, Caltech concurrent computation program, 1989.
23. *Faye A.* Solving the Aircraft Landing Problem with time discretization approach // European Journal of Operational Research. — 2015. — Vol. 242, N 3. — P. 1028—1038.

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

Вересников Георгий Сергеевич — канд. техн. наук, ст. науч. сотрудник, ✉ veresnikov@mail.ru,

Егоров Николай Александрович — канд. техн. наук. ст. науч. сотрудник, ✉ egorov@ipu.ru,

Кулида Елена Львовна — канд. техн. наук, ст. науч. сотрудник, ✉ lenak@ipu.ru,

Лебедев Валентин Григорьевич — д-р техн. наук, уч. секретарь, ✉ lebedev@ipu.ru,

Институт проблем управления им. В.А. Трапезникова РАН, г. Москва.