УДК 004.896 DOI 10.18522/2311-3103-2025-4-21-37

М.А. Ганжур, Б.К. Лебедев, О.Б. Лебедев

МНОГОСТАДИЙНЫЙ МУРАВЬИНЫЙ АЛГОРИТМ ОДНОМЕРНОЙ УПАКОВКИ НА БАЗЕ ЭФФЕКТИВНЫХ МЕТОДОВ КОДИРОВАНИЯ РЕШЕНИЙ, И ДВУХУРОВНЕВОЙ ЭВОЛЮЦИОННОЙ ПАМЯТИ

Целью работы является разработка и исследование методов биоинспирированного поиска для решения задач одномерной упаковки в одинаковые контейнеры на базе эффективных алгоритмов кодирования и декодирования решений, композитного критерия и двухуровневой структуры эволюционной памяти. В работе предложена структура упорядоченного кода упаковки одномерных элементов в одинаковые контейнеры главное достоинство которого заключается в том, что одному решению упаковки соответствует один код и наоборот. Поисковая процедура базируется на модифицированной метаэвристике муравьиного алгоритма. На каждой итерации алгоритм одномерной упаковки имеет многостадийную структуру. Стадии выполняются последовательно одна за другой, начиная с первой. Каждая стадия C_k включает процедуры, выполняемые агентом z_k. Число стадий равно числу агентов в популяции плюс заключительная стадия итерации. Основная задача, решаемая конструктивным алгоритмом на стадии C_k, заключается в построении кода R_k упаковки множества элементов X в одинаковые контейнеры. Стадия делится на периоды по числу формируемых агентом z_k списков $X_{jк}$. Период делится на этапы. На каждом периоде последовательно по этапам решаются следующие задачи: агент z_k конструктивным алгоритмом формирует набор R_k упорядоченных списков $X_{jк}$ одномерной упаковки в одинаковые контейнеры; рассчитываются оценки f_{jk} упаковки каждого контейнера O_j элементами списка $<X_{jк}>$; рассчитывается количество λ_{jk} феромона, пропорциональное оценке f_{jk}; рассчитывается оценка $\Omega_k=\sum_i(f_{jk})$ одномерной упаковки множества элементов X в H одинаковых контейнеров; производится отложение феромона на ребрах графа G, соответствующих списку $X_{jк}$ в ячейки накопительной матрицы памяти E второго уровня. После формирования всеми агентами z_k популяции Z упорядоченных списков R_k, накопленный феромон добавляется в основную матрицу памяти Ф первого уровня. Для каждого R_k рассчитывается общий показатель F_k качества упаковки множества элементов X. Заключительная операция на итерации – испарение феромона на ребрах графа G и фиксация z_k с лучшим F_k. Проведены экспериментальные исследования заключающиеся в выяснении качества работы метода на тестовых наборах большой размерности. Для сравнения разработанного алгоритма с известными методами и с приближенными алгоритмами авторами было выбрано несколько групп бенчмарок из различных источников.

Одномерная упаковка; одинаковые контейнеры; методология; упорядоченный код решения; эволюционная память; двухуровневый композитный критерий; поисковая оптимизация; декомпозиция; роевой интеллект; муравьиная колония.

M.A. Ganzhur, B.K. Lebedev, O.B. Lebedev

MULTI-STAGE ANT ALGORITHM OF ONE-DIMENSIONAL PACKING BASED ON EFFICIENT DECISION ENCODING METHODS AND TWO-LEVEL EVOLUTIONARY MEMORY

The aim of the work is to develop and study bioinspired search methods for solving problems of one-dimensional packaging in identical containers based on effective algorithms for encoding and decoding solutions, composite criteria and a two-level structure of evolutionary memory. The paper proposes the structure of an ordered code for packing one-dimensional elements into identical containers, the main advantage of which is that one packaging solution corresponds to one code and vice versa. The search procedure is based on the modified metaheuristics of the ant algorithm. At each iteration, the one-dimensional packing algorithm has a multistep structure. The stages are performed sequentially one after the other, starting from the first one. Each stage of the C_k includes procedures performed by the z_k agent. The number of stages is equal to the number of agents in the population plus the final iteration stage. The main task solved by the constructive algorithm at the C_k stage is to construct the R_k code for packing a set of X elements into identical containers. The stage is divided into periods according to the number of lists X_{jk} generated by the agent z_k. The period is divided into stages. In each period, the following tasks are solved sequentially in stages: agent z_k constructively generates a set Rk of ordered lists Xjk of one-

dimensional packaging in identical containers; fjk estimates of the packaging of each container O_j by elements of the list $<X_{jk}>$ are calculated; the amount of λ_{jk} pheromone proportional to the f_{jk} estimate is calculated; the estimate $\Omega_k=\sum_i(f_{jk})$ is calculated one-dimensional packing of a set of elements X into H identical containers; pheromone is deposited on the edges of graph G corresponding to the list X_{jk} in the cells of the accumulative memory matrix E of the second level. After all agents of the z_k population Z have formed ordered lists of R_k, the accumulated pheromone is added to the main memory matrix Φ of the first level. For each R_k, the total F_k indicator of the packaging quality of the set of X elements is calculated. The final operation in the iteration is pheromone evaporation on the edges of graph G and fixation of z_k with the best F_k. Experimental studies have been conducted to determine the quality of the method's operation on large-dimensional test sets. To compare the developed algorithm with known methods and approximate algorithms, the authors selected several groups of benchmarks from various sources.

One-dimensional packaging; identical containers; methodology; ordered solution code; evolutionary memory; two-level composite criterion; search engine optimization; decomposition; swarm intelligence; ant colony.

Введение. Задача одномерной упаковки в одинаковые контейнеры является распространенной производственной задачей. Эта задача является членом большой семьи задач, многие из которых естественно возникают на практике и состоят в разбиении множества элементов на непересекающиеся подмножества [1]. Например, задачей одномерной упаковки является такая важная проблема исследования операций, как проблема машинного планирования, загрузки грузовиков с ограничением по весу. Она решается при производстве стали, стекла, бумаги, дизайне СБИС, составлении бюджета, форматировании таблиц, постраничном разбиении, а также при упаковке мусора в минимальное количество мусорных ведер и т.д. [2]. Рассматриваемая задача упаковки одномерных элементов в контейнеры является *NP*-полной [3]. Несмотря на высокую изученность задачи, существование огромного количества различных методов ее решения для ряда контрольных (тестовых) задач оптимальное решение не получено [4]. Более того, на данный момент не существует представленного в литературе универсального алгоритма, способного одинаково эффективно решать все тестовые задачи. В настоящее время весьма актуальными являются проблемы создания и разработки новых высокоточных и быстрых методов для синтеза проектных решений.

Возникшие потребности в решении задач большой и очень большой размерности является побудительным мотивом исследований и разработок новых эффективных алгоритмов. Ввиду вышеизложенного, задача упаковки блоков является актуальной проблемой комбинаторной оптимизации, стоящей перед специалистами в различных областях производства.

Анализ литературы показывает, что наиболее успешными в этих условиях являются математические методы, в которых заложены принципы природных механизмов принятия решений [5]. К таким методам можно отнести, прежде всего, методы адаптивного поведения [6], метод эволюционного моделирования [7], гибридные модели [8–1], алгоритмы роевого интеллекта [11, 12] и муравьиные алгоритмы (ACO) [13, 14].

На основе модернизированной метаэвристики муравьиного алгоритм была разработана иерархическая структура алгоритма. Одним из новых направлений непрекращающегося поиска наиболее эффективных методов упаковки стало использование с одной стороны бионических методов и алгоритмов, а с другой стороны использование эффективных методов кодирования решений и реконфигурируемой архитектуры, что позволило значительно уменьшить временную сложность алгоритма упаковки и увеличило на порядок размерность решаемых задач.

На основе модернизированной метаэвристики муравьиного алгоритм разработана иерархическая структура алгоритма одномерной упаковки в одинаковые контейнеры.

Постановка задачи одномерной упаковки в одинаковые контейнеры. Дано множество элементов $X=\{x_i|i=1,2,...,n_x\}$. Вес элементов задается множеством $W=\{w_i|i=1,2,...,n\}$. Необходимо упаковать все элементы в контейнеры, т.е. множество X разбить на H непустых и непересекающихся подмножеств X_j. Для всех формируемых подмножеств $X=\cup X_j$, $X_j\neq\varnothing$, $(\forall i,j)[X_i \cap X_j=\varnothing]$.

Каждое подмножество элементов X_j помещается в контейнер Q_j. $Q=\{Q_j|j=1,2,...,H\}$ – множество контейнеров, в которые распределено множество элементов X.

Пусть имеется некоторое решение задачи одномерной упаковки.

Суммарный вес W_j подмножества элементов X_j, назначенных в контейнер Q_j.

$$W_j \sum w_i, \text{ для } \{i|x_i \in X_j\}. \quad (1)$$

Задается максимально допустимый суммарный вес D элементов, назначенных в каждый контейнер Q_j: $W_j \le D$. Обозначим как $\delta_j = D - W_j$ остаток (незаполненный объем) контейнера Q_j. Тогда $\Delta = \sum_j \delta_j$ – суммарный остаток объема (веса) множества контейнеров Q.

Цель оптимизации – минимизация числа контейнеров H.

Для увеличения однородности ландшафта целевой функции и уменьшения резких перепадов ее значений в процессе поиска в пространстве решений авторами предложен двухуровневый композитный критерий F, включающий два показателя: (H – число контейнеров), (Δ – суммарный остаток веса H контейнеров). Критерий оптимизации:

$$F = \alpha H + \beta \Delta, \quad (2)$$

где α и β – коэффициенты пропорциональности.

Цель оптимизации – минимизация F.

В отличие от большинства критериев многокритериальной оптимизации, показатели H и Δ композитного критерия F не находятся в состоянии конфликта. Напротив, улучшение значения одного из них способствует улучшению значения другого.

Рассмотрим стандартную процедуру упаковки [2].

Дан упорядоченный, некоторым образом, список $<X>$ элементов для упаковки.

Первый элемент x_1 списка $<X>$ включается в контейнер Q_1. Элементы $x_2, \ldots, x_n$ рассматриваются в порядке их индексов: рассматриваемый элемент упаковывается в текущий контейнер Q_j, если не происходит переполнения контейнера; в противном случае он упаковывается в новый контейнер Q_{j+1}, который становится текущим. Временная сложность алгоритма $O(n)$. Очевидно, что существует такая последовательность элементов в списке, при которой решение задачи упаковки будет оптимальным. Таким образом, решение задачи упаковки сводится к нахождению упорядоченного списка $<X>$, используемого стандартной процедурой упаковки.

В результате декодирования, с помощью рассмотренной выше процедуры, список X представляется в виде упорядоченной последовательности списков X_j. $<X> = <X_1, X_2, X_3, \ldots, X_H>$, где X_j множество элементов, упакованных в контейнер Q_j.

Декодирование производится путем последовательного просмотра кода решения (вектора X), в соответствии с вышеприведенной процедурой.

Отметим, что порядок, в котором расположены элементы, входящие в состав одного контейнера Q_j, а также порядок, в котором списки X_j расположены в векторе $<X>$ не изменяет сущности решения.

Недостаток этого подхода заключается в том, что перестановка пары вершин, входящих в один список X_j, и перестановка списков X_j в одном списке X приводит к увеличению пространства поиска и к повышенным затратам при поиске решения.

Пусть имеется некоторое решение задачи одномерной упаковки: список $<X> = <X_1, \ldots, X_H>$, $<X> = \cup X_j$, $X_j \neq \varnothing$, $|X_j| = n_j$, $(\forall i,j)[X_i \cap X_j = \varnothing]$. Элементы каждого списка X_j упакованы в одном контейнере Q_j. Поскольку порядок расположения сформированных подмножеств X_j в составе списка $<X>$, и порядок, в котором элементы каждого подмножества X_j, расположенные в контейнере Q_j, не имеют значения, то одному коду решения будет соответствовать число списков P, равное $(H!) \cdot \Pi_j(n_j!)$, где $(H!)$ – количество комбинаций взаимного расположения списков X_j в составе общего списка $<X>$.

$n_j!$ – количество комбинаций взаимного расположения n_j элементов списка X_j.

В связи с этим при поиске решения актуальна проблема синтеза и просмотра только одного списка, соответствующего одному конкретному решению (одному коду).

Структура упорядоченного кода. В работе предложена структура упорядоченного кода $<R> = <<X_1>, <X_2>, \ldots, <X_H>>$. $R = \cup X_j$, $X_j \neq \varnothing$, $|X_j| = n_j$, $(\forall i,j)[X_i \cap X_j = \varnothing]$ упаковки одномерных элементов в одинаковые контейнеры главное достоинство которого заключается в том, что одному решению упаковки соответствует один код и наоборот.

Упорядоченный список $<R>$ обладает следующими свойствами:

- Пусть A_j первый элемент списка $<X_j>$.
- Первый элемент A_1 списка $<R>$имеет максимальный вес среди всех элементов списка $<R>$.
- Списки $<X_j>$в составе общего списка R упорядочены таким образом, что первые элементы A_j всех списков $<X_j>$ упорядочены в составе списка R по убыванию веса.
- В каждом списке $<X_j>$ элементы x_i, начиная со второго, упорядочены по убыванию веса.
- Построение упорядоченного списка $<R>$, обладающего перечисленными выше свойствами, выполняется последовательно на базе опорного упорядоченного вектора X^* с динамически изменяемым составом.
- Элементы исходного опорного списка X для упаковки упорядочиваются по убыванию весов, т.е. $w_{i-1} > w_i$. X^* – упорядоченное множество X. Например: $X^*=<x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}>$.

$$W=<8, 7, 6, 6, 5, 4, 4, 3, 3, 3, 2, 1>.$$

- Формирование упорядоченного списка элементов $<R>$ производится по очереди путем последовательного формирования списков $<X_j>$.
- Очередной, начиная с первого, список $<X_j>$ элементов формируется путем последовательного заполнения контейнера Q_j до упора.
- Формирование каждого списка $<X_j>$ включает две стадии.

Назовем разницу между предельным весом контейнера и суммарным весом упакованных в контейнере элементов остатком δ_j. Обозначим, как X^*_T и R_T текущие в процессе поиска состояния (состав) векторов X^* и $<R>$. Начальные текущие состояния $X^*_T=<x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}>$. $R_T = \varnothing$.

На <u>первом шаге S_1</u> этапа формирования очередного списка $<X_j>$ выбирается первый $A1$, от начала, ненулевой элемент x^* опорного вектора X^*_T. $A1=x^*$. Определяется вес w^* элемента x^*. Элемент x^* размещается в первой позиции списка $<X_j>$и исключается из списка X^*_T. Рассчитывается вес остатка $\delta_j=D-w^*$ объема контейнера Q_j.

На <u>втором шаге этапа формирования</u> $<X_j>$ формируется список X_O элементов x_i таких, что каждый из них может быть добавлен в контейнер Q_j (в состав списка $<X_j>$) без его переполнения, т.е. $w_i \le \delta_j$. Далее на каждом из последующих шагов, начиная с первого, случайным образом, выбирается элемент $x_i \in X_O$, который включается в список $<X_j>$ и список R_T, а затем исключается из списка X^*_T. Рассчитывается остаток $\delta_j=O_j-w^*$ контейнера Q_j, формируется новый список X_O.

Если $X_O \neq \varnothing$, то повторяются действия, связанные с выбором элемента $x_i \in X_O$, для включения его в список X_j и список R_T.

Если $X_{O=}\varnothing$, то процесс заполнения контейнера Q_j (формирования списка X_j) завершается.

Ниже приведен пример формирования списков $<X_j>$ на базе описанных выше векторов X^*_T и W.

Список обозначений.

$X=<x_1, x_2, \ldots, x_n>$ – исходный список элементов.

X^* – упорядоченный по убыванию веса опорный список элементов множества X.

$<X_j>$ – упорядоченный код упаковки контейнера O_j.

$<R_k>$ – построенный агентом z_k упорядоченный код упаковки множества элементов X в одинаковые контейнеры, X^*_T и R_T текущие состояния векторов X^* и R.

$\Theta=\{R_k|i=1,2,\ldots,n_k\}$ – набор кодов, построенных популяцией агентов Z на итерации $Z=\{z_k|k=1,2,\ldots,n_z\}$ – популяция агентов.

k – номер агента.

l – номер итерации.

Пусть $X^*=<x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}>$.

$W=<8, 7, 6, 6, 5, 4, 4, 3, 3, 3, 2, 1>$.

$D=13$.

(j=1).
S_1. Выбор 1 элемента $A_1=x_1$ в X^*_T. w_1=8.
Расчет остатка $O_1=D - w_1$=13-8=5.
Формирование X_O=<$x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}$>.
S_2. Случайный выбор элемента x_i в X_O: x_5.
w_5=5, $O_2 = O_1-w_5$ =5-5=0. δ_j =0.
Включение x_1, x_5 в R_T= <x_1, x_5 >.
Удаление x_1, x_5 из X^*_T. $j=j+1$.
X^*_T= <$x_2, x_3, x_4, x_6, x_7, x_8, x_9, x_{10}, x_{11}, _{12}$>.

(j=2).
S_1. Выбор 1 элемента $A_j=x_2$ в X_T. w_2=7.
Расчет остатка $O_1=D-w_2$=13-7=6.
Формирование X_O=<$x_3, x_4, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}$>.
S_2. Случайный выбор элемента x_i в X_O: x_3.
w_3=6, $O_2 = O_1-w_3$ =6-6=0. δ_j =0
Включение x_2, x_3 в R_T=<x_1, x_5, x_2, x_3 >.
Удаление x_2, x_3 из X_T. $j=j+1$.
X^*_T = <$x_4, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}$>.

(j=3).
S_1. Выбор 1 элемента $A_j= x_4$ в X^*_T. w_4=6.
Расчет остатка $O_1=D-w_4$=13-6=7.
X_O=<$x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}$>.
S_2. Случайный выбор элементов x_i в X_O: x_8, x_6.
w_8=3, $O_2 = O_1-w_8$ =7-3=4.
w_6=4, $O_3 = O_2-w_6$= 4.4=0. δ_j =0.
$H=x_4, x_8, x_6$. Упорядоченный H=<x_4, x_6, x_8>.
Включение H в R_T=<$\underline{x_1}, x_5, \underline{x_2}, x_3, \underline{x_4}, x_6, x_8$>.
Удаление x_4, x_6, v_8 из X^*_T. $j=j+1$.
X^*_T= <$x_7, x_9, x_{10}, x_{11}, x_{12}$>.

(j=4).
S_1. Выбор 1 элемента $A1= x_7$ в X^*_T. w_7=4.
Расчет остатка $O_1=D - w_7$=13-4=9.
X_O=< $x_9, x_{10}, x_{11}, x_{12}$>.
S_2. Случайный выбор элементов в X_O: $x_{10}, x_9, x_{12}, x_{11}$.
w_{10}=3, $O_2 = O_1-w_{10}$=9-3=6.
w_9=3, $O_3 = O_2-w_9$ =6-3=3. w_{12}=1, $O_4 = O_3$-1=3-1=2.
w_{12}=1, $O_4 = O_3$-1=3-1=2.
w_{11}=2, $O_5 = O_4-w_{11}$=2-2=0. δ_j =0.
$H= x_7, x_{10}, x_9, x_{12}, x_{11}$. Упорядоченный H= <$x_7, x_9, x_{10}, x_{11}, x_{12}$>.
Включение H в R_T.
R_T=<$\underline{x_1}, x_5, \underline{x_2}, x_3, \underline{x_4}, x_6, x_8, \underline{x_7}, x_9, x_{10}, x_{11}, x_{12}$>.
Удаление $x_7, x_9, x_{10}, x_{11}, x_{12}$ из X^*_T.
X^*_T= <>. $R=R(j)$. δ_j=0.
Если $X^*_T=\varnothing$, то конец формирования упорядоченного списка.
Сформированный код имеет вид:
R=< $\underline{x_1}, x_5, \underline{x_2}, x_3, \underline{x_4}, x_6, x_8, \underline{x_7}, x_9, x_{10}, x_{11}, x_{12}$ >.
W=<$\underline{8}$ 5 $\underline{7}$ 6 $\underline{6}$ 4 3 $\underline{4}$ 3 3 2 1>.
X_1=<$\underline{8}$ 5>, X_2=<$\underline{7}$, 6>, X_3=<$\underline{6}$, 4, 3>, X_4=<$\underline{4}$, 3, 3, 2, 1>. δ=0.

Решение задачи одномерной упаковки в одинаковые контейнеры методом муравьиной колонии. В качестве модели пространства поиска решений задачи одномерной упаковки в одинаковые контейнеры используется полный граф $G(X,U)$, где U – множество всех ребер полного графа, связывающих множество вершин x, соответствующих множеству элементов. $X=\{x_i|i=1,2,...,n_x\}$. $U=\{u_{iv}|i=1,2,...,n_x; v=1,2,...,n_u; n_x=n_u\}$. u_{iv} – ребро графа G, связывающее вершины x_i и x_v. $(\forall_i \forall_v)[(x_i \in X)\&(x_v \in X)\&(u_{iv} \in U]$.

Задача одномерной упаковки сводится к задаче разбиения множества вершин X полного графа решений $G(X,U)$ на минимальное число H упорядоченных подмножеств $X_j \subset X$, с ограничениями на суммарный вес вершин каждого X_j. $W_j \leq D$. W_j, фактически является суммарным весом элементов, упакованных в контейнер.

Основной процедурой, при **формировании упорядоченного кода** R является процедура формирования упорядоченного списка X_j элементов, упаковываемых в одном контейнере O_j.

Каждый агент z_k популяции Z формирует упорядоченные списки X_{jk}, входящие в состав кода упаковки R_k, по очереди. Авторами предлагаются три подхода к решению поставленной задачи.

Первый подход предусматривает формирование, на базе списков W и X, общего упорядоченного кода $<R>$ упаковки путем последовательного формирования по очереди, начиная с первого, упорядоченных списков X_j, с минимизацией показателя $\delta_j = D - W_j$, $\delta_j \geq 0$, $X = \cup X_j$, $X_j \neq \varnothing$, $(\forall i,j)[X_i \cap X_j = \varnothing]$. δ_j – остаток (не заполненная часть контейнера O_j). $\delta_j \rightarrow min$. j – порядковый номер в очереди списков X_j.

Критерий оптимизации задачи упаковки – показатель H, находится во взаимной зависимости (корреляционной связи) с результатами формирования упорядоченных списков X_j, определяется как сформированное число H списков X_j.

Второй подход заключается в последовательном формировании упорядоченного кода общего решения с минимизацией показателя H – числа контейнеров.

Третий подход заключается в формировании упорядоченного кода общего решения с иерархической минимизацией H путем формирования последовательности упорядоченных списков X_j с минимизацией показателя δ_j. Процесс поиска решений муравьиным алгоритмом итерационный.

На каждой итерации алгоритм упаковки имеет многостадийную структуру. Стадии выполняются последовательно одна за другой, начиная с первой (рис. 1). Каждая стадия C_k включает процедуры, выполняемые агентом z_k. Число стадий равно числу агентов в популяции плюс заключительная стадия итерации. Основная задача, решаемая конструктивным алгоритмом на стадии C_k, заключается в построении кода R_k упаковки множества элементов X в одинаковые контейнеры. На каждой стадии C_k формируется набор $R_k = \{X_{jk} | j = 1,2,..n_j\}$ упорядоченных кодов списков $X_{j\kappa}$ вершин графа G. Стадия делится на периоды (рис. 2) по числу n_j формируемых агентом z_k списков $X_{j\kappa}$. Период делится на этапы (рис. 3).

$\{И_l\} \rightarrow \{C_1, C_2, \ldots, C_k, \ldots, C_{закл}\}$; $C_k \rightarrow \{П_{1k}, П_{2k}, \ldots П_{jk}, \ldots\}$; $П_{jk} \rightarrow \{Э_{1jk}, Э_{2jk}, Э_{3jk}\}$.

И – итерация; С – стадия; П – период; Э – этап.

На каждом периоде $П_{j\kappa}$ последовательно по этапам решаются следующие задачи: агент z_k конструктивным алгоритмом формирует набор R_k, $|R_k| = H$ упорядоченных списков $X_{j\kappa}$ одномерной упаковки в одинаковые контейнеры; рассчитываются оценки f_{jk} упаковки каждого контейнера O_j элементами списка $<X_{j\kappa}>$; рассчитывается количество λ_{jk} феромона, пропорциональное оценке f_{jk}; рассчитывается оценка $\Omega_k \sum_i (f_{jk})$ одномерной упаковки множества элементов X в H одинаковых контейнеров; производится отложение феромона на ребрах графа G, соответствующих списку $X_{j\kappa}$ в ячейки накопительной матрицы *памяти* $E = ||\varepsilon_{iv}||_{n \times n}$ второго уровня. После формирования всеми агентами z_k популяции Z упорядоченных списков R_k феромон, накопленный в $E = ||\varepsilon_{iv}||_{n \times n}$, добавляется в основную матрицу памяти $\Phi = ||\varphi_{iv}||_{n \times n}$ первого уровня.

Для каждого R_k рассчитывается общий показатель F_k качества упаковки множества элементов $X = \{x_i | i = 1,2,\ldots,n\}$.

Заключительная операция на итерации – испарение феромона на ребрах графа G и фиксация z_k с лучшим F_k.

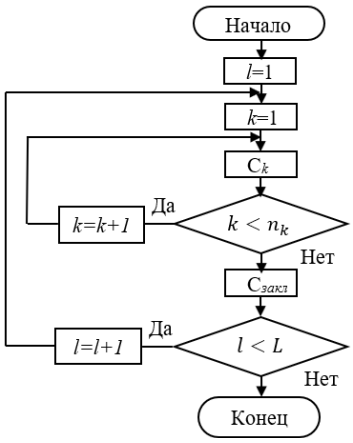


Рис. 1. Схема реализации блока процедур «Стадия»

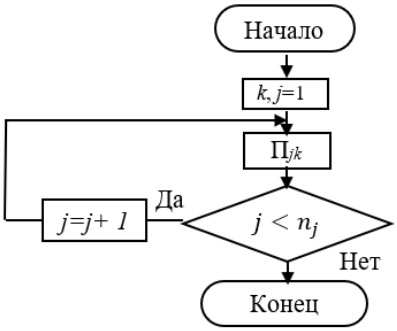


Рис. 2. Схема реализации блока «Период»

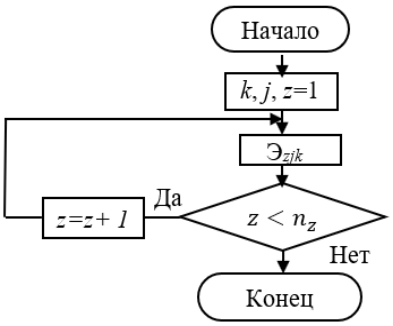


Рис. 3. Схема реализации блока «Этап»

Алгоритм одномерной упаковки на основе метода муравьиной колонии. Структурная схема алгоритма одномерной упаковки на основе метода муравьиной и представлена на рис. 4.

Фиксируются исходные данные.

$X=\{x_i|i=1,2,\ldots,n\}$ – исходный список элементов.

Параметры:

n_l – число итераций;

n_k – число агентов популяции;

D – предельный вес контейнера;

$W=\{w_i|i=1,2,\ldots,n\}$ – вес элементов.

Формируется начальное состояние базы данных муравьиного алгоритма.

В соответствии с методикой, вышеизложенной на примере, случайным образом формируется исходное множество $\Theta=\{R_k|k=1,2,\ldots,n_k\}$ упорядоченных кодов упаковки. Рассчитываются оценки решений, соответствующих сформированным кодам. Для каждого решения упаковки, соответствующего коду $R_к$, рассчитывается количество феромона, которое откладывается в ячейках основной матрицы памяти $\Phi=||\varphi_{iv}||_{n\times n}$.

Агенты обладают памятью.

Приводится в начальное состояние память агента z_k:

Элементам накопительной матрицы $E=||\varepsilon_{iv}||_{n\times n}$, присваивается нулевое значение.

$j=1$. $X^*_T=X^*$. $R_T=\varnothing$. $X_j=\varnothing$. X^*_T

В начале каждой итерации l формируется: опорный упорядоченный вектор $X^*=<x_i|i=1,2,\ldots,n|w_i\geq w_{i+1}>$. $k=1$. На каждой итерации l на стадии C_k агентом z_k последовательно решается задача формирования набора $R_k=\{X_{jk}|j=1,2,..n_j\}$ упорядоченных списков X_{jk} каждый из которых является кодом упаковки соответствующего контейнера O_j.

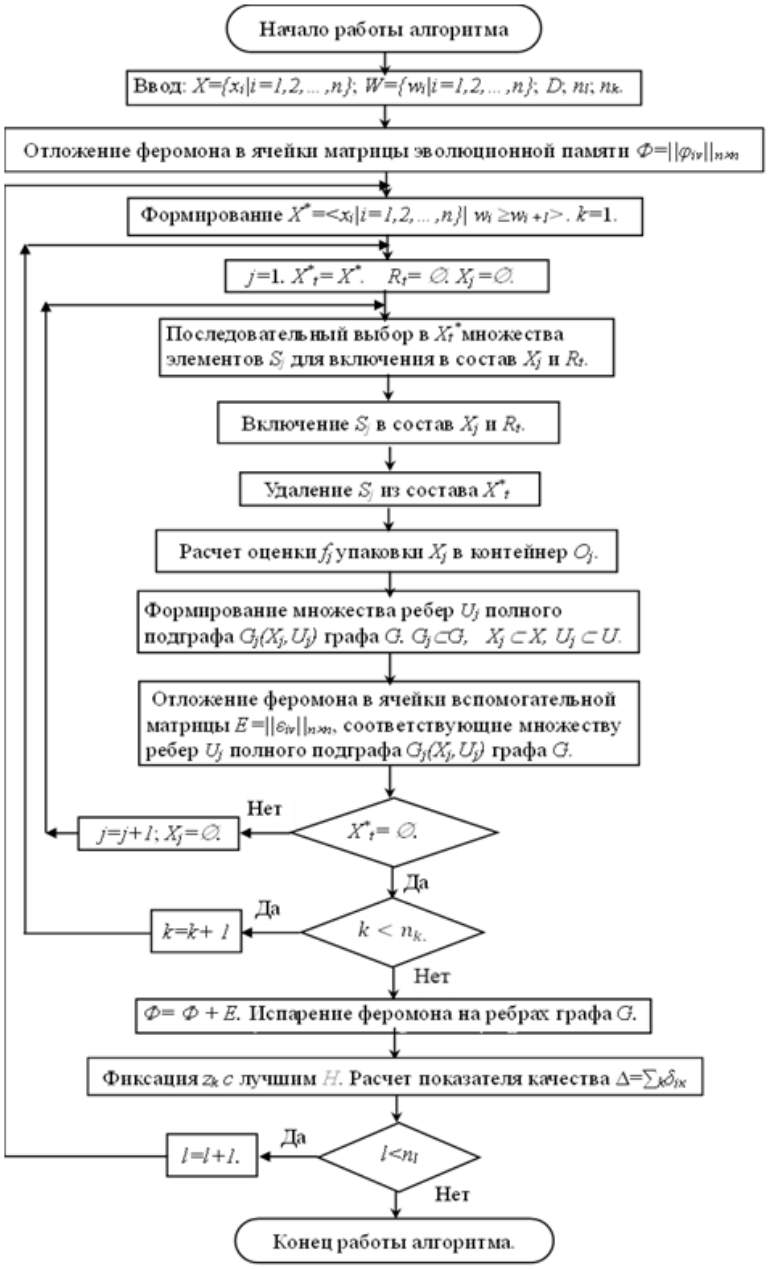


Рис. 4. Структурная схема алгоритма одномерной упаковки в одинаковые контейнеры

Задача формирования упорядоченного списка X_{jk}, являющегося кодом упаковки контейнера O_j, решается агентом z_k на каждом периоде $П_{jk}$, стадии $С_k$. Число списков равно числу периодов.

Необходимо упаковать элементы в минимальном количестве H_k контейнеров, т.е. множество X разбить на H_k непустых и непересекающихся подмножеств X_{jk} таких, что каждое подмножество элементов X_{jk} помещается в контейнер Q_j без переполнения.

Отметим, что число H_k контейнеров определяется только после того, как все элементы будут распределены по контейнерам.

Каждый список $<X_{jk}>$ формируется путем последовательной упаковки O_j.

Поиск решения задачи упаковки контейнера O_j (формирование списка X_{jk}) осуществляется агентом z_k на полном графе $G(X,U)$, где X – множество вершин, соответствующих множеству элементов. $|X|=n$, U –множество ребер графа G.

Моделирование поведения агентов в задаче упаковки контейнера связано с использованием эволюционной памяти, фиксируемой двумя матрицами памяти $\Phi=||\varphi_{iv}||_{n\times n}$ и $E=||\varepsilon_{iv}||_{n\times n}$.

Отражение коллективной эволюционной памяти в течение жизни популяции Z агентов осуществляется путем «откладывания» феромона на множестве ребер U графа G.

На начальном этапе во всех ячейках матрицы Φ откладывается одинаковое (небольшое) количество феромона Φ/v, где $v=|U|$. Параметр Φ задается априори.

Существующая каноническая парадигма муравьиного алгоритма предполагает построение маршрута в графе поиска решений G. В отличие от канонической парадигмы авторами предлагается концепция, заключающаяся в том, что целью агента z_k является формирование на графе поиска решений $G(X,U)$ набора R_k упорядоченных списков X_{jk}, отвечающих требованиям: элементы каждого списка упорядочены по убыванию веса w_i вершин множества X_{jk}; первый элемент множества X_{jk} имеет максимальное значение; суммарный вес W_{jk} множества вершин X_{jk} меньше D, $W_{jk} \le D$; а остаток $\delta_{jk}=(D-W_{jk})$ имеет минимальное значение. $W_{jk} \sum w_i$ для $\{i|x_i \in X_{jk}\}$, $X_{jk} \subset X$, $|X_{jk}|=n_{jk}$.

Формирование упорядоченных списков. На первом этапе $Э_{1jk}$ очередного периода $П_{jk}$ агент z_k конструктивным алгоритмом формирует упорядоченный список $<X_{jk}>$ вершин графа G. Каждый список $<X_{jk}>$ включает элементы, для упаковки в контейнере O_j.

Последовательное формирование упорядоченного списка $<X_{jk}>$ производится по шагам, путем включения на каждом шаге t элемента $x_i \in X_O(t)$ в Q_j.

Пусть X^*_T – множество вершин, соответствующих еще не упакованным элементам.

На первом шаге $t=1$, выбирается первый ненулевой элемент x_i в списке X^*_T, который включается в состав формируемых списков $<X_{jk}>$ и R_T и исключается из списка X^*_T.

На каждом последующем шаге, начиная с $t=2$ определяется множество $e_k(t)$ элементов x_i, включенных в состав $<X_{jk}(t)>$.

Определяется суммарная стоимость $W_{jk}(t)$ элементов множества $e_k(t)$.

Формируется множество вершин $X_O(t) \subset X^*_T$, соответствующих еще не упакованным элементам, таких, что если $x_i \in X_O(t)$, то элемент x_i может быть помещен в контейнер Q_j без переполнения, т.е. без превышения ограничения на вес. $(W_{jk}(t)+w_i) \le D$.

Если $X_0(t) \neq \varnothing$, то агент z_k просматривает в качестве кандидатов все вершины $x_i \in X_0(t)$.

Для каждой вершины $x_i \in X_0(t)$ определяется множество ребер $U_{ie} \subset U$, связывающих x_i с множеством элементов $e_k(t)$. Рассчитывается параметр $\xi_{ik}(t)$ – суммарный уровень феромона в ячейках матрицы памяти $\Phi=||\varphi_{iv}||_{n\times n}$, соответствующих ребрам $U_{ie} \subset U$ графа G, связывающих x_i с вершинами множества $e_k(t)$.

Вероятность P_{ijk} включения вершины $x_i \in X_0(t)$ в список $X_{jk}(t)$ определяется следующим соотношением:

$$P_{ijk}=\xi_{ik}(t) / \Sigma_i(\xi_{ik}(t)) \ (i|x_i \in X_0(t)). \quad (3)$$

Агент z_k с вероятностью P_{ijk} выбирает одну из вершин $x_i \in X_0(t)$, которая включается в список $<X_{jk}(t)>$. $<X_{jk}(t+1)>=<X_{jk}(t) \cup x_i>$. $t=(t+1)$.

Для сформированного списка $<X_{jk}>$ рассчитывается остаток $\delta_{jк}$ упаковки контейнера O_j и оценка f_{jk} упаковки O_j множеством элементов списка $<X_{jk}>$.

$$f_{jk}=\alpha \cdot \delta_{jк} / D, \quad (4)$$

где α – коэффициент пропорциональности.

Организация эволюционной памяти. Запоминание оценки решения соответствует процедуре отложения феромона на модели (ребрах графа) в количестве пропорциональном оценке решения. В работе используется циклический (ant-cycle) метод муравьиных систем. В этом случае феромон в полном объеме откладывается на ребрах графа G после полного формирования решений всеми агентами на каждой итерации [13]. Для хранения

интегральных оценок решений используется двухуровневая структура эволюционной памяти в виде двух матриц: *накопительная* $E=||\varepsilon_{iv}||_{n\times n}$, и *основная* матрицы памяти $\Phi=||\varphi_{iv}||_{n\times n}$.

На каждой итерации в процессе работы алгоритма агентами для последовательного накопления феромона за итерацию используется матрица $E=||\varepsilon_{iv}||_{n\times n}$, где ε_{iv} суммарное количество феромона, отложенного всеми агентами за одну итерацию на ребре u_{iv} графа G. В начале каждой итерации элементам накопительной матрицы $E=||\varepsilon_{iv}||_{n\times n}$, присваивается нулевое значение.

Для хранения феромона, накопленного всеми агентами на итерации в процессе работы алгоритма, используется матрица памяти $\Phi=||\varphi_{iv}||_{n\times n}$, где φ_{iv} количество феромона, отложенного агентами на ребре u_{iv} графа G. $(\forall_i \forall_v)[(x_i \in X)\&(x_v \in X)\&(u_{iv} \in U]$.

На втором этапе $Э_{2jk}$ периода $П_{jk}$ стадии C_k текущей итерации на множестве вершин $<X_{jk}(t)>$ графа G формируется полный подграф $G_{jk}=(X_{jk}, U_{jk})$, и определяется множество U_{jk} ребер $u_{iv} \in U_{jk}$ подграфа G_{jk}. $G_{jk} \subset G$, $<X_{jk}> \subset X$, $U_{jk} \subset U$. $u_{jv} \in U_{jk}$.

На третьем этапе $Э_{3jk}$ периода $П_{jk}$ рассчитывается количество λ_{jk} феромона, пропорциональное оценке f_{jk} упаковки списка $<X_{jk}>$.

$$\lambda_{jk}=\beta \cdot Qf_{jk}. \quad \lambda_{jk}=\beta \cdot \Phi / f_{jk}, \tag{5}$$

где β – коэффициент пропорциональности.

Φ – общее количество феромона, откладываемое агентом z_k на ребрах полного подграфа G_{jk}, f_{jk} – оценка списка $<X_{jk}>$, полученного агентом z_k на l-ой итерации. Чем меньше f_{jk}, тем больше феромона откладывается на ребрах U_{jk} подграфа G_{jk} и, следовательно, тем больше вероятность выбора этих ребер при формировании $<X_{jk}>$ на следующей итерации.

Феромон в количестве λ_{jk} откладывается в ячейках накопительной матрицы $E=||\varepsilon_{iv}||_{n\times n}$, соответствующих множеству U_{jk} ребер полного подграфа $G_{jk}=(X_{jk}, U_{jk})$ графа G.

$$(\forall u_{iv})[u_{iv} \in U_{jk})\&[\varepsilon_{iv}=\varepsilon_{iv}+\lambda_{jk}].$$

Каждый раз после формирования агентом z_k упорядоченного списка $<X_{jk}>$ производится расчет параметров δ_{jk}, f_{jk}, λ_{jk}, после чего на каждом ребре полного подграфа $G_{jk}=(X_{jk}, U_{jk})$ графа G «откладывается» феромон в количестве λ_{jk}.

В работе используются три методики учета результатов решения задачи упаковки.

При первом методе.

Агент на первом этапе после формирования каждого $<X_{jk}>$, каждый раз после формирования агентом z_k $<X_{jk}>$:

- рассчитываете оценка f_{jk} упаковки контейнера O_j элементами множества $<X_{jk}>$;
- рассчитывается количество $\lambda_{jk}=\beta \cdot Q/f_{jk}$ феромона, пропорциональное оценке f_{jk} упаковки списка $<X_{jk}>$;
- определяется множество U_{jk} ребер $u_{iv} \in U_{jk}$ полного подграфа $G_{jk}=(X_{jk}, U_{jk})$ графа G, $G_{jk} \subset G$, построенного на множестве вершин $<X_{jk}>$.
- феромон в количестве λ_{jk} откладывается в ячейках накопительной матрицы $E=||\varepsilon_{iv}||_{n\times n}$, соответствующих множеству U_{jk} ребер полного подграфа $G_{jk}=(X_{jk}, U_{jk})$ графа G.

$$(\forall u_{iv})[u_{iv} \in U_{jk})\&[\varepsilon_{iv}=\varepsilon_{iv}+\lambda_{jk}].$$

При втором методе после формирования на итерации l агентом z_k упорядоченного списка $R_k=\{X_{jk}|j=1,2,..,n_j\}$:

- фиксируется число $H=n_j$ упакованных контейнеров;
- формируется множество ребер $U_k=\cup U_j$;
- определяется количество феромона $\varphi_\kappa=\xi H$, которое откладывается на ребрах полных подграфов G_{jk}.

В работе предагаются три подхода учета результатов решения задачи упаковки.

При первом подходе феромон откладывается совместно по первой и второй методикам.

При втором подходе феромон откладывается только по первой методике.

При третьем подходе феромон откладывается только по второй методике.

После формирования на итерации l агентом z_k упорядоченного списка $R_k=\{X_{jk}|j=1,2,..n_j\}$ фиксируется число $H=n_j$ упакованных контейнеров определяется количество феромона $\varphi_к=\xi H$, которое дополнительно откладывается на ребрах полных подграфов G_{jk}.

Для хранения феромона, накопленного агентами на итерации в процессе работы алгоритма, используется матрица памяти $\Phi=||\varphi_{iv}||_{n\times n}$, где φ_{iv} количество феромона, отложенного агентами на ребре u_{iv} графа G. $(\forall_i \forall_v)[(x_i \in X)\&(x_v \in X)\&(u_{iv} \in U]$.

После формирования на итерации всеми агентами z_k популяции Z упорядоченных списков $\Theta=\{R_k|k=1,2,\ldots,n_k\}$ феромон, накопленный в $E=||\varepsilon_{iv}||_{n\times n}$, добавляется в основную матрицу памяти $\Phi=||\varphi_{iv}||_{n\times n}$. $\forall u_{iv}[\varphi_{iv}=\varphi_{iv}+\varepsilon_{iv}]$.

Для каждого R_k рассчитывается общий показатель качества $\Delta_k=\sum_j \delta_{jк}$ упаковки множества элементов $X=\{x_i|i=1,2,\ldots,n\}$ и фиксация z_k с лучшим H.

На заключительной стадии итерации осуществляется испарение феромона на ребрах графа G в соответствии с формулой:

$$\forall u_{iv}[\varphi_{iv}=\varphi_{iv}(1-\rho)], \tag{6}$$

где ρ – коэффициент обновления.

Далее осуществляется переход на следующую итерацию.

Заключительная операция на итерации – испарение феромона на ребрах графа G (в соответствующих ячейках памяти $\Phi=||\varphi_{iv}||_{n\times n}$). После выполнения всех действий на итерации находится агент с лучшим решением, которое запоминается.

Временная сложность этого алгоритма зависит от времени жизни колонии l (число итераций), количества вершин графа n и числа муравьев m, и определяется как $O(l\cdot n^2\cdot m)$.

Второй и третий подходы отличаются способами учета критериев оптимизации. При втором подходе, как и при первом минимизируется суммарный остаток $\Delta_k=\sum_j \delta_{jк}$ упаковки множества элементов $X=\{x_i|i=1,2,\ldots,n\}$ и минимизируется число контейнеров H. Производится двойное отложение феромона. Первое связано с отложением феромона на ребрах каждого подграфа $G_{jk,}$ построенного на множестве вершин $<X_{jk}>$ в количестве пропорциональном остатку $\delta_{jк}$. Второе связано с отложением феромона в количестве пропорциональном числу контейнеров H на ребрах полного подграфа G_k, построенного на множестве вершин $R_k=\{X_{jk}|j=1,2,..n_j\}$ – кода упаковки, построенного. агентом z_k.

Экспериментальные исследования. Для получения и исследования оптимальных и квазиоптимальных решений задачи одномерной упаковки был разработан программный комплекс. Основным способом исследования является запуск приложения на известных тестовых примерах – бенчмарках.

В окне, показанном на рис. 5, настраиваются параметры муравьиного алгоритма: число итераций, число муравьев, коэффициент испарения, начальный феромон и максимальный феромон. Степень завершения тестирования отображается Δ в окне, изображенном на рис. 6.

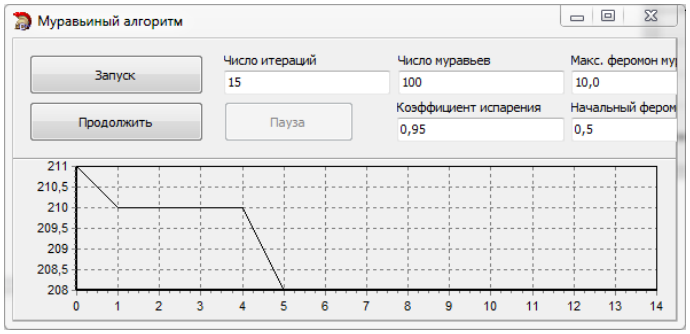


Рис. 5. Окно муравьиного алгоритма

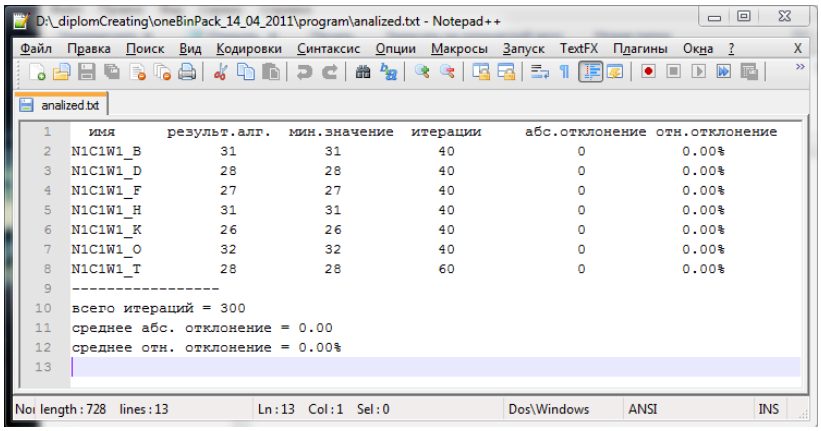


Рис. 6. Окно результатов тестирования

Временная сложность для алгоритма в целом составляет $O(m \cdot l \cdot n^2)$, где n – число вершин (элементов), m – число муравьев, l – время жизни колонии.

На рис. 7 представлена экспериментальная зависимость времени работы алгоритма от числа элементов n. По оси времени – время работы для 8 муравьев и 1 итераций.

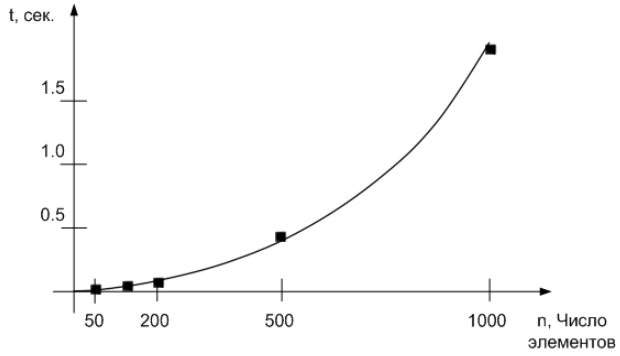


Рис. 7. Зависимость времени работы алгоритма от числа элементов

В качестве исследуемых параметров были выбраны следующие параметры муравьиного алгоритма: число муравьев m, коэффициент испарения k, отношение начального уровня феромона к максимальному феромону муравья γ. На рис. 8 представлены экспериментальные графики. Испытания проводились на группе равномерных бенчмарок, описанных в работе Э. Фалкенауэра [15].

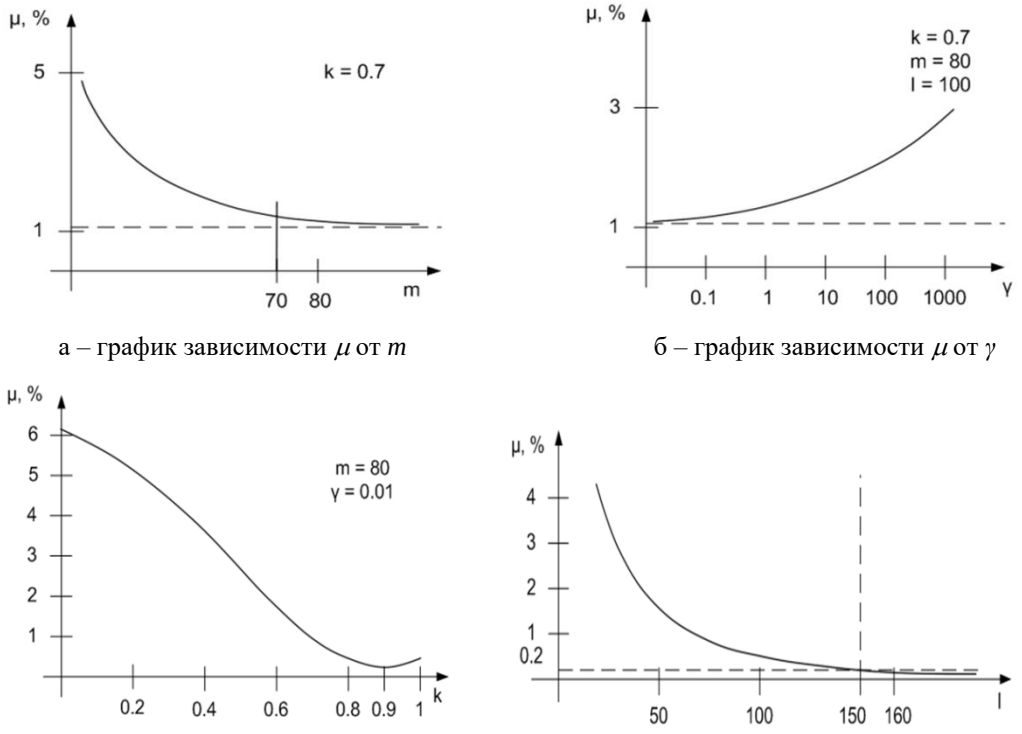
По оси ординат всех графиков (рис. 8) отложена величина μ – это среднее относительное отклонение от минимального значения качества решения.

$$\mu = \frac{1}{B}\sum_{i=0}^{B}\frac{x_i - x_{\text{мин}}}{x_{\text{мин}}} \cdot 100\%. \quad (7)$$

График на рис. 8,а построен при коэффициенте испарения k .7. Из него видно, что наименьшее количество муравьев, которое позволяет алгоритму находить качественное решение, находится в пределах 7 –80. γ – это отношение начального уровня феромона к максимальному значению феромона муравья.

Оптимальное значение коэффициента испарения равно .9, это видно на графике, изображенном на рис. 4,в. Этот график построен при m 8 и γ=0.01.

При найденных оптимальных параметрах был построен график сходимости, изображенный на рисунке 4г. Из него видно, что решение, отличающееся от минимального не более чем на ,2%, находится в среднем за 15 -16 итераций.



а – график зависимости μ от m

б – график зависимости μ от γ

в – график зависимости μ от k

г – график сходимости – зависимости μ от l

Рис. 8. Исследование параметров

Для сравнения разработанного алгоритма с другими известными методами авторами было выбрано несколько групп бенчмарок из различных источников: H1, H2, H3, U120, U250, U500, U1000, T60, T120, T249, T501 [15].

В табл. 1 АВР сравнивается с алгоритмом BISON [16]. Сравниваемые параметры:

- «Nмин.» – число бенчмарок в наборе, для которых найдено минимальное решение;
- «Δ(%)» – среднее отклонение результата от лучшего значения;
- $T_{ср}$ – среднее время, затраченное на одну бенчмарку.

Как видно, разработанный алгоритм эффективнее почти по всем параметрам, проигрывает только по времени выполнения набора данных H1.

Таблица 1

Сравнение с алгоритмом BISON

Тестовый набор	АВР			BISON		
	мин.	Δ(%)	$T_{ср}$	Nмин.	Δ (%)	$T_{ср}$
H1 (720)	709	1,57	43	597	2,38	32
H2 (480)	458	2,1	15	273	4,94	16
H3 (10)	4	4,68	15	1	5,85	18

В табл. 2 алгоритм АВР сравнивается с HGGA [15] и MTP [17]. В сравнении с MTP алгоритм АВР работает намного лучше как по времени, так и по качеству. Муравьиный алгоритм оказался быстрее генетического алгоритма HGGA, хотя немного уступает ему по качеству.

Таблица 2

Сравнение с алгоритмами HGGA и MTP

Тестовый набор	ABP		HGGA		MTP	
	Δ (%)	$T_{ср}$	Δ(%)	$T_{ср}$	Δ(%)	$T_{ср}$
U120 (20)	0,2	30	2,3	31	2,5	30
U250 (20)	2,2	37	2,5	33	2,9	29
U500 (20)	0,4	33	2,8	35	3,1	30,5
U1000 (20)	0,5	29	3.0	39	3,4	23
T60 (20)	1,1	36	3,2	37	4.0	31
T120 (20)	0,4	33	2.5	29	3,3	23
T249 (20)	0,7	31	2.4	28	3,0	28
T501 (20)	1,2	30	2.4	36	2,9	26

Проведено сравнение разработанного алгоритма с приближенными алгоритмами *FFD, BFD, WFD* для наборов данных H1, H2 и H3 [17].

Все три приближенных алгоритма уступают муравьиному по качеству, хотя позволяют получать решение за более короткий промежуток времени. При найденных оптимальных параметрах был построен график сходимости. Решение, отличающееся от минимального не более чем на ,2%, находится в среднем за 15 -16 итераций.

В результате экспериментов было установлено, что показатели качества ABP имеют более высокие значения чем в работах, представленных в литературе [18–20]. По сравнению с существующими алгоритмами достигнуто улучшение результатов на 3-5%.

Временная сложность алгоритма ABP, полученная экспериментальным путем, практически совпадает с теоретическими исследованиями и для рассмотренных тестовых задач составляет (BCA ≈ $O(n^2)$).

Заключение. Исходя из анализа решений, получаемых алгоритмами упаковки, построенными на различных метаэвристиках и эвристиках, было установлено, что большинство из них может быть улучшено. На основании этих выводов была разработана новая композитная поисковая архитектура алгоритма для решения задачи одномерной упаковки в одинаковые контейнеры.

В работе предложена структура упорядоченного кода <*R*> упаковки одномерных элементов в одинаковые контейнеры главное достоинство которого заключается в том, что одному решению упаковки соответствует один код и наоборот.

Разработан алгоритм построения агентом $z_{k.}$ упорядоченного кода упаковки множества элементов *X* в одинаковые контейнеры. Это позволило значительно уменьшить временную сложность алгоритма упаковки и увеличило на порядок размерность решаемых задач, что востребовано при решении распределительных задач. Для хранения интегральных оценок решений используется двухуровневая структура эволюционной памяти в виде двух матриц: *накопительная* $E=||\varepsilon_{iv}||_{n\times m}$, и *основная* матрицы памяти $\Phi=||\varphi_{iv}||_{n\times m}$.

Наличие двухуровневой структуры эволюционной памяти позволяет распараллеливать поиск решения. Агенты роя на каждой итерации параллельно формируют решения, опираясь на фиксированную основную память Накопленную информацию, отражающую оценки решений и степени приспособленности значений управляющих операторов, агенты вносят в основную память после выполнения итерации.

На основе модернизированной метаэвристики муравьиного алгоритма разработана иерархическая реконфигурируемая структура алгоритма одномерной упаковки в одинаковые контейнеры. Процесс поиска решений муравьиным алгоритмом итерационный. На каждой итерации алгоритм упаковки имеет многостадийную структуру. Стадии выполняются последовательно одна за другой, начиная с первой. Каждая стадия C_k включает процедуры, выполняемые агентом z_k. Число стадий равно числу агентов в популяции плюс заключительная стадия итерации. Основная задача, решаемая конструктивным алгоритмом на стадии C_k, заключается в построении кода R_k упаковки множества элементов *X* в одинаковые контейнеры. На каждой стадии C_k формируется набор R_k упорядоченных

списков $X_{jк}$ вершин графа G. Стадия делится на периоды по числу n_j формируемых агентом z_k списков $X_{jк}$. Период делится на этапы. Такой подход позволяет производить реконфигурацию архитектуры алгоритма с учетом специфики решаемой задачи.

Для увеличения однородности ландшафта целевой функции и уменьшения резких перепадов ее значений в процессе поиска в пространстве решений авторами предложен двухуровневый композитный критерий F, включающий два показателя: (H-число контейнеров), (Δ-суммарный остаток веса контейнеров). Критерий оптимизации – $F=\alpha H+\beta\Delta$, где α и β – коэффициенты пропорциональности. Цель оптимизации – минимизация F. В отличие от большинства критериев многокритериальной оптимизации, показатели H и Δ композитного критерия F не находятся в состоянии конфликта. Напротив, улучшение значения одного из них способствует улучшению значения другого.

Для сравнения разработанного алгоритма с известными методами авторами использовались несколько групп бенчмарок. В сравнении с поисковыми алгоритмами BISON, HGGA и MTP алгоритм ACO работает лучше как по времени, так и по качеству.

Все три приближенных алгоритма *FFD, BFD, WFD* уступают муравьиному по качеству, хотя позволяют получать решение за короткий промежуток времени.

По сравнению с существующими алгоритмами достигнуто улучшение результатов на 3-5%.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Кормен Т.Л.* Алгоритмы. Построение и анализ. – М.: Вильямс, 2007. – 196 с.
2. *Bischoff E.E.* Cutting and packing // European Journal of Operational Research. – 1995. – P. 503-505.
3. *Лебедев Б.К., Лебедев О.Б., Ганжур М.А.* Одномерная упаковка модифицированным методом муравьиной колонии // Тр. конгресса по интеллектуальным системам и информационным технологиям «IS– IT'23». Научное издание. – Таганрог: Изд-во ЮФУ, 2023. – Т. 1. – С. 153-161.
4. *Курейчик В.М., Потарусов Р.В.* Проблема одномерной упаковки элементов // Известия ТРТУ. – 2006. – № 8. – С. 88-93.
5. *Карпенко А.П.* Современные алгоритмы поисковой оптимизации. Алгоритмы, вдохновленные природой: учеб. пособие. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2021. – 448 с.
6. *Курейчик В.М., Лебедев Б.К., Лебедев О.Б.* Поисковая адаптация: теория и практика. – М.: Физматлит, 2006. – 272 с.
7. *Бекетов С.М., Зубкова Д.А., Гинцяк А.М., Бурлуцкая Ж.В., Редько С.Г.* Современные методы оптимизации и особенности их применения // Russian Technological Journal. – 2025. – 13 (4). – P. 78-94. – https://doi.org/10.32362/2500-316X-2025-13-4-78-94. – EDN: CVZOXD.
8. *Gupta J.N., Ho J.C.* A New Heuristic Algorithm for the One-dimensional Bin-packing Problem // Production Planning & Control. – 2009. – Vol. 10. – P. 598-603.
9. *Levine J.M., Ducatelle F.C.* Ant Colony Optimization and Local Search for Bin Packing and Cutting Stock Problems // Centre for Intelligent Systems and their Applications. – Edinburgh: University of Edinburgh, 2003. – P. 50-64.
10. *Лебедев Б.К., Лебедев О.Б.* Распределение ресурсов на основе гибридных моделей роевого интеллекта // Научно-технический вестник информационных технологий, механики и оптики. – 2017. – Т. 17, № 6. – С. 1063-1073.
11. *Engelbrecht A.P.* Fundamentals of Computational Swarm Intelligence. – Chichester: John Wiley & Sons, 2005. – 114 p.
12. *Лебедев Б.К., Лебедев О.Б., Лебедева Е.О.* Роевой алгоритм планирования работы многопроцессорных вычислительных систем // Электронный научный журнал «Инженерный вестник Дона». – № 3. – 2017.
13. *Лебедев О.Б.* Модели адаптивного поведения муравьиной колонии в задачах проектирования. – Таганрог. Изд-во ЮФУ, 2013. – 199 с.
14. *Dorigo M.A., Stutzle T.M.* Ant colony optimization. – Cambridge: MIT Press, 2004. – 151 с.
15. *Falkenauer E.I.* A hybrid grouping genetic algorithm for bin packing // Journal of Heuristics. – 1996. – Vol. 2. – P. 5-30.
16. *Scholl A.D., Klein R.B., Jurgens C.F.* BISON: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem // Computers and Operations Research. – 1997. – P. 627-645.
17. *Martello S.K., Toth P.M.* Bin-packing problem // Algorithms and Computer Implementations. – 2002. – P. 221-245.

18. *Raidl G.R.* A Unified View on Hybrid Metaheuristics // In: Lecture Notes in Computer Science. – Springer, Verlag, 2006. – P. 1-12.
19. *Cong J., Romesis M., Xie M.* Optimality, Scalability and Stability Study of Partitioning and Placement Algorithms // Proc. of the International Symposium on Physical Design. – Monterey, CA, 2004. – P. 88-94.
20. *Лебедев Б.К., Лебедев О.Б., Ганжур М.А.* Биоинспирированный алгоритм распределения блоков по контейнерам // Известия ЮФУ. Технические науки. – 2024. – № 4. – С. 14-30.

REFERENCES

1. *Kormen T.L.* Algoritmy. Postroenie i analiz [Algorithms. Construction and analysis]. Moscow: Vil'yams, 2007, 196 p.
2. *Bischoff E.E.* Cutting and packing, *European Journal of Operational Research*, 1995, pp. 503-505.
3. *Lebedev B.K., Lebedev O.B., Ganzhur M.A.* Odnomernaya upakovka modifitsirovannym metodom murav'inoy kolonii [One-dimensional packing by a modified ant colony method], *Tr. kongressa po intellektual'nym sistemam i informatsionnym tekhnologiyam «IS– IT'23»* [Proceedings of the Congress on Intelligent Systems and Information Technologies «IS-IT'23»]. Scientific publication. Taganrog: Izd-vo YuFU, 2023, Vol. 1, pp. 153-161.
4. *Kureychik V.M., Potarusov R.V.* Problema odnomernoy upakovki elementov [The problem of one-dimensional packing of elements], *Izvestiya TRTU* [Izvestiya TSURE], 2006, No. 8, pp. 88-93.
5. *Karpenko A.P.* Sovremennye algoritmy poiskovoy optimizatsii. Algoritmy, vdokhnovlennye prirodoy: ucheb. posobie [Modern algorithms of search engine optimization. Algorithms inspired by nature: a tutorial]. Moscow: Izd-vo MGTU im. N.E. Baumana, 2021, 448 p.
6. *Kureychik V.M., Lebedev B.K., Lebedev O.B.* Poiskovaya adaptatsiya: teoriya i praktika [Search adaptation: theory and practice]. Moscow: Fizmatlit, 2006, 272 p.
7. *Beketov S.M., Zubkova D.A., Gintsyak A.M., Burlutskaya Zh.V., Red'ko S.G.* Sovremennye metody optimizatsii i osobennosti ikh primeneniya [Modern optimization methods and their application features], *Russian Technological Journal*, 2025, 13 (4), pp. 78-94. Available at: https://doi.org/10.32362/2500-316X-2025-13-4-78-94. EDN: CVZOXD.
8. *Gupta J.N., Ho J.C.* A New Heuristic Algorithm for the One-dimensional Bin-packing Problem, *Production Planning & Control*, 2009, Vol. 10, pp. 598-603.
9. *Levine J.M., Ducatelle F.C.* Ant Colony Optimization and Local Search for Bin Packing and Cutting Stock Problems, *Centre for Intelligent Systems and their Applications*. Edinburgh: University of Edinburgh, 2003, pp. 50-64.
10. *Lebedev B.K., Lebedev O.B.* Raspredelenie resursov na osnove gibridnykh modeley roevogo intellekta [Resource allocation based on hybrid models of swarm intelligence], *Nauchno-tekhnicheskiy vestnik informatsionnykh tekhnologiy, mekhaniki i optiki* [Scientific and technical bulletin of information technologies, mechanics and optics], 2017, Vol. 17, No. 6, pp. 1063-1073.
11. *Engelbrecht A.P.* Fundamentals of Computational Swarm Intelligence. Chichester: John Wiley & Sons, 2005, 114 p.
12. *Lebedev B.K., Lebedev O.B., Lebedeva E.O.* Roevoy algoritm planirovaniya raboty mnogoprotsessornykh vychislitel'nykh sistem [Swarm algorithm for scheduling the work of multiprocessor computing systems], *Elektronnyy nauchnyy zhurnal «Inzhenernyy vestnik Dona»* [Electronic scientific journal «Engineering Bulletin of the Don»], No. 3, 2017.
13. *Lebedev O.B.* Modeli adaptivnogo povedeniya murav'inoy kolonii v zadachakh proektirovaniya [Models of adaptive behavior of an ant colony in design problems]. Taganrog. Izd-vo YuFU, 2013, 199 p.
14. *Dorigo M.A., Stutzle T.M.* Ant colony optimization. Cambridge: MIT Press, 2004, 151 p.
15. *Falkenauer E.I.* A hybrid grouping genetic algorithm for bin packing, *Journal of Heuristics*, 1996, Vol. 2, pp. 5-30.
16. *Scholl A.D., Klein R.B., Jurgens C.F.* BISON: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem, *Computers and Operations Research*, 1997, pp. 627-645.
17. *Martello S.K., Toth P.M.* Bin-packing problem, *Algorithms and Computer Implementations*, 2002, pp. 221-245.
18. *Raidl G.R.* A Unified View On Hybrid Metaheuristics, *In: Lecture Notes In Computer Science*. Springer, Verlag, 2006. – P. 1-12.
19. *Cong J., Romesis M., Xie M.* Optimality, Scalability and Stability Study of Partitioning and Placement Algorithms, *Proc. of the International Symposium on Physical Design*. Monterey, CA, 2004, pp. 88-94.
20. *Lebedev B.K., Lebedev O.B., Ganzhur M.A.* Bioinspirirovannyy algoritm raspredeleniya blokov po konteyneram [Bioinspired algorithm for distributing blocks among containers], *Izvestiya YuFU. Tekhnicheskie nauki* [Izvestiya SFedU. Engineering Sciences], 2024, No. 4, pp. 14-30.

Ганжур Марина Александровна – Донской государственный технический университет; e-mail: mganzhur@yandex.ru; г. Ростов-на-Дону, Россия; тел.: 89081819111; кафедра вычислительных систем и информационной безопасности; старший преподаватель.

Лебедев Борис Константинович – Южный федеральный университет; e-mail: lebedev.b.k@gmail.com; г. Таганрог, Россия; тел.: 89282897933; кафедра систем автоматизированного проектирования им. В.М. Курейчика; д.т.н.; профессор.

Лебедев Олег Борисович – МИРЭА – Российский технологический университет; e-mail: lebedev.ob@mail.ru; г. Москва, Россия; тел.: 89085135512; кафедра информатики; д.т.н.; профессор.

Ganzhur Marina Aleksandrovna – Don State Technical University; e-mail: mganzhur@yandex.ru; Rostov-on-Don, Russia; phone:+79081819111; the Department of Computing Systems and Information; security senior lecturer.

Lebedev Boris Konstantinovich – Southern Federal University; e-mail: lebedev.b.k@gmail.com; Taganrog, Russia; phone: +79282897933; the Department of Computer Aided Design named after V.M. Kureichik; dr. of eng. sc.; professor.

Lebedev Oleg Borisovich – MIREA – Russian University of Technology; e-mail: lebedev.ob@mail.ru; Moscow, Russia; phone: +79085135512; the Department of Computer Science; dr. of eng. sc.; professor.

УДК 621.004.42 DOI 10.18522/2311-3103-2025-4-37-46

И.Е. Моисеенко, С.П. Тарасов, И.И. Турулин

АНАЛИЗ УСТОЙЧИВОСТИ И ОСОБЕННОСТЕЙ ПРАКТИЧЕСКОЙ РЕАЛИЗАЦИИ ФИЛЬТРОВ ХОГЕНАУЭРА КАК РЕКУРСИВНЫХ ЦИФРОВЫХ ФИЛЬТРОВ С КОНЕЧНОЙ ИМПУЛЬСНОЙ ХАРАКТЕРИСТИКОЙ

Рассматриваются вопросы устойчивости каскадных интегратор-гребенчатых (CIC – cascade integrator-comb) фильтров, используемых в цифровой обработке сигналов, в том числе для децимации и интерполяции. Проведен краткий обзор современных публикаций, касающихся архитектурной оптимизации CIC-фильтров. Основное внимание уделено повышению устойчивости фильтров к переполнению разрядной сетки, анализу их устойчивости и методу синтеза рекурсивных КИХ-фильтров (фильтров с конечной импульсной характеристикой). Для лучшего понимания природы устойчивости CIC-фильтров в работе приведены математические выкладки, иллюстрирующие особенности накопления постоянной составляющей при различных конфигурациях блоков. Предложено изменение структуры CIC-фильтра, заключающееся в перестановке блоков интегратора и гребенчатого фильтра. Доказано, что такое изменение предотвращает накопление постоянной составляющей сигнала в интеграторах и, следовательно, исключает переполнение разрядной сетки вследствие накопления постоянной составляющей в интеграторе. Этот подход базируется на свойстве линейных фильтров, согласно которому изменение порядка включения не влияет на передаточную функцию. амплитудно-частотную характеристику, но в случае цифровых реализаций позволяет существенно снизить вероятность переполнения. Возможности аппаратной и программной реализации таких структур рассматриваются с точки зрения минимизации потерь точности и увеличения надежности работы систем цифровой обработки сигналов. Предложено использование целых чисел или чисел с фиксированной точкой для устранения накопления ошибок квантования. Кроме того, была разработана программа на языке Python, реализующая CIC-фильтр с учетом устойчивости к постоянной составляющей во входном сигнале и точным выполнением операций. Полученные результаты сопоставлены с современными подходами, представленными в научных исследованиях последних лет. Предложенные решения могут быть полезны при разработке цифровых фильтров для систем с ограниченными вычислительными ресурсами и повышенными требованиями к стабильности.

CIC-фильтр; Хогенауэр; цифровая обработка сигналов; децимация; импульсная характеристика; амплитудно-частотная характеристика; фазочастотная характеристика; Python; устойчивость фильтров.