УДК 519.854.3:681.5.01
DOI: 10.26102/2310-6018/2026.59.8.001

Метод перехода к постоянной длине вектора параметров целевой функции в задачах оптимизации работы мостовых кранов на складе слябов

И.С. Щеголихин✉, С.М. Андреев

Магнитогорский государственный технический университет им. Г.И. Носова, Магнитогорск, Российская Федерация

Резюме. Ряд оптимизационных задач, возникающих при рационализации логистики склада непрерывнолитых заготовок металлургического предприятия, оперирует вектором параметров целевой функции переменной длины. Это накладывает ограничения на выбор методов, применимых для решения таких задач. Ряд существующих методов приведения вектора параметров целевой функции к постоянному размеру, которые применяются для комбинаторной оптимизации, обладают выразительной ограниченностью, т. е. применимы лишь к узкоспециализированному классу задач. Целью настоящего исследования является приведение вектора параметров целевой функции, возникающей в задачах оптимизации работы мостовых кранов на складе слябов, к постоянному размеру, что позволит для решаемой оптимизационной задачи использовать все известные методы оптимизации из подходящего класса, а также понизит ресурсоемкость алгоритма оптимизации за счет сокращения количества параметров целевой функции. В данной работе авторы предлагают метод, являющийся обобщением L-систем и клеточного кодирования, позволяющий для заданной целевой функции перейти от вектора параметров переменного размера к вектору параметров постоянного размера. Новизной работы является обобщение идей, специфичных для узкоспециализированных областей научного знания, на множество классов оптимизационных задач, в частности на задачи составления расписания для мостовых кранов. Авторами рассмотрена целевая функция, возникающая при совместном решения задач составления расписания мостовых кранов и выбора слябов для графика горячей прокатки. Для этой целевой функции был применен предложенный метод: описаны функции смены поискового пространства в виде простых алгоритмов. Результатом явилось сокращение размерности векторов целевой функции минимум в 10 раз. Предложенный метод может быть использован при проектировании цифрового двойника склада непрерывнолитых заготовок, а также для управления иерархией команд: от самых простых (для управления отдельными компонентами кранов) до самых сложных (задающих логистические операции над слябами).

Ключевые слова: вектор параметров, целевая функция, слябы, клеточное кодирование, непрямое кодирование, прямое кодирование.

Для цитирования: Щеголихин И.С., Андреев С.М. Метод перехода к постоянной длине вектора параметров целевой функции в задачах оптимизации работы мостовых кранов на складе слябов. *Моделирование, оптимизация и информационные технологии.* 2026;14(8). URL: https://moitvivt.ru/ru/journal/article?id=2423 DOI: 10.26102/2310-6018/2026.59.8.001

A method for transitioning to a fixed-length parameter vector of the objective function in optimization problems of overhead crane operations in a slab yard

I.S. Shchegolikhin✉, S.M. Andreev

G.I. Nosov Magnitogorsk State Technical University, Magnitogorsk, the Russian Federation

Abstract. A number of optimization problems that arise when rationalizing the warehouse logistics of a continuously cast slab yard of a metallurgical enterprise operate with a variable-length vector of parameters of the objective function. This circumstance, consequently, restricts the choice of methods applicable to such problems. Moreover, an increase in the parameter vector of the objective function in many cases raises the resource intensity of the optimization algorithm. Several existing methods that reduce the parameter vector of the objective function to a constant size and that find use in combinatorial optimization suffer from a pronounced limitation: they apply only to a highly specialized class of problems. The present study aims to reduce the parameter vector of the objective function that appears in problems of optimizing the work of bridge cranes in a slab yard to a constant size. Such a reduction will allow one to use, for the optimization problem under consideration, all known optimization methods from the appropriate class and, in addition, will lower the resource intensity of the optimization algorithm by decreasing the number of parameters of the objective function. In this work, the authors propose a method that generalizes L-systems and cellular coding and makes it possible, for a given objective function, to move from a variable-size parameter vector to a constant-size parameter vector. The novelty of the work consists in extending ideas that are specific to highly specialized fields of scientific knowledge to a set of classes of optimization problems, in particular to scheduling problems for bridge cranes. The authors examine an objective function that arises when one jointly solves bridge crane scheduling problems and the problem of selecting slabs for a hot rolling schedule. They apply the proposed method to this objective function and describe the functions that change the search space in the form of simple algorithms. As a result, the dimensionality of the objective function vectors decreases at least by a factor of 10. The proposed method can be used in the design of a digital twin for a continuous casting billet storage yard, as well as for managing a hierarchy of commands: from the simplest (for controlling individual crane components) to the most complex (defining logistics operations involving slabs).

Keywords: parameter vector, objective function, slabs, cellular encoding, indirect encoding, direct encoding.

For citation: Shchegolikhin I.S., Andreev S.M. A method for transitioning to a fixed-length parameter vector of the objective function in optimization problems of overhead crane operations in a slab yard. *Modeling, Optimization and Information Technology*. 2026;14(8). (In Russ.). URL: https://moitvivt.ru/ru/journal/article?id=2423 DOI: 10.26102/2310-6018/2026.59.8.001

Введение

Склад слябов металлургического предприятия является важным связующим звеном между двумя технологическими процессами: непрерывной разливкой стали, осуществляемой при помощи машины непрерывного литья заготовок (МНЛЗ), и горячей прокаткой на стане. Склад необходим для временного хранения слябов, поступающих от МНЛЗ, а также для согласования темпов производства. Мостовые краны перемещают слябы на склад, по складу и со склада. На склад слябы поступают в заранее известной последовательности, называемой графиком непрерывного литья. Со склада слябы извлекаются также в заранее известной последовательности, которая задается графиком горячей прокатки. График горячей прокатки состоит из последовательностей элементов. Элементы графика горячей прокатки задают требуемые характеристики для сляба, подлежащего прокатке. Как правило, выбор для каждого элемента графика ведется среди нескольких слябов. Последовательность, в которой расположены элементы графика горячей прокатки, задает порядок отправки слябов, соответствующих этим элементам, на прокатку [1, 2].

На складе слябов естественным образом возникает проблема составления расписания кранов (CSP – Crane Scheduling Problem), которая заключается в определении команд, которые должны быть выполнены конкретным мостовым краном [3]. Также на складе слябов возникает задача выбора слябов (SSP – Stack Shuffling Problem), которая заключается в выборе конкретных слябов на складе для графика горячей прокатки (один

сляб на один элемент графика) [4]. Также при поступлении слябов на склад должна быть решена задача распределения мест хранения (SLAP – Storage Location Assignment Problem), при решении которой каждому слябу, пришедшему от МНЛЗ, назначается место в штабеле на складе [5]. Каждая из этих задач формулируется как смешанная целочисленная программа [6, 7]. При извлечении слябов решение CSP напрямую зависит от решения SSP.

Задачи CSP, SLAP и SSP могут решаться как совместно, так и отдельно. В любом случае необходимо использовать методы оптимизации нулевого порядка. Кроме того, алгоритм оптимизации должен обладать гибкостью для учета сложных ограничений, нацеленных на соблюдение: безопасного расстояния между мостовыми кранами; минимальной высоты, на которую должен быть поднят краном транспортируемый сляб; максимально допустимой высоты штабеля, в который укладываются слябы на складе; строгой последовательности извлечения слябов и др. [8].

Мостовые краны могут выполнять только множество простых команд (назовем их атомарными командами), среди них: $o_1(x)$ – переместить балку крана к координате x; $o_2(x)$ – переместить каретку крана к координате x; $o_3(x)$ – опустить лебедку крана на расстояние x; $o_4(x)$ – поднять лебедку крана на расстояние x; o_5 – захватить сляб, расположенный непосредственно под крюком; $o_6(x)$ – освободить захваченный сляб; $o_7(x)$ – ожидать x секунд в текущей позиции.

Слябы укладываются в штабеля один поверх другого, отсюда следует, что мостовые краны могут захватить только самый верхний сляб. Для извлечения мостовыми кранами сляба, который расположен в штабеле i-м по счету снизу-вверх, необходимо сначала совершить $n - i$ операций перекладываний вышележащих слябов (n – общее количество слябов в штабеле), и только за ними последует операция извлечения i-го сляба. Это в свою очередь означает, что для извлечения каждого заданного сляба со склада необходимо формировать последовательность атомарных команд переменной длины (для каждого сляба длина последовательности может отличаться). Также если в одной складской области работает более одного крана, то один кран в некоторых ситуациях вынужден ожидать завершения выполнения команды другого крана, что также обуславливает переменную длину вектора параметров, которая может сильно увеличиваться в зависимости от того, сколько слябов расположены над извлекаемой заготовкой.

Переменная длина векторов параметров целевой функции ограничивает множество доступных для решения оптимизационной задачи методов, а вместе с тем рост размерности этих векторов практически для всех оптимизационных алгоритмов вызывает рост их ресурсоемкости.

Подходящими кандидатами для решения подобного рода задач являются эволюционные вычисления. Эволюционные вычисления активно применяются в качестве методов оптимизации нулевого порядка [9]. Среди них можно выделить: генетические алгоритмы (ГА), алгоритм имитации отжига, генетическое программирование, эволюционное программирование и др. Кроме того, отдельные методы эволюционных вычислений комбинируются учеными, образуя новые варианты, как, например, в работе [10]. При помощи эволюционных вычислений можно, в частности, отыскивать экстремальные значения функций нескольких вещественных аргументов [11, 12].

Большинство методов оптимизации, основанных на эволюционных вычислениях, способны работать с целевой функцией любой природы, что делает их применение возможным для задач, которые могут быть сведены к модели целочисленного или смешанного целочисленного программирования, как, например, составление

расписаний [13, 14], задача коммивояжера [15, 16] и другие (в том числе SLAP, SSP и CSP).

При решении задач оптимизации целевой функции при помощи эволюционных вычислений необходимо каким-то образом закодировать элементы $dom(F)$, чтобы эволюционный алгоритм смог работать с закодированными элементами. В генетических алгоритмах элементы называются фенотипами, а их закодированное представление для использования генетическим алгоритмом называют генотипом. Многие исследователи применяют данную терминологию и для других методов, основанных на эволюционных вычислениях, этого соглашения авторы настоящей статьи будут придерживаться далее.

Выделяют два основных типа кодирования фенотипов в генотип: прямое и непрямое кодирования. При прямом кодировании (direct encoding) структурная единица фенотипа отображается непосредственно в одну структурную единицу генотипа – в ген. Примером прямого кодирования может служить представление фенотипа (A, B, C, D), из множества всех перестановок на множестве $\{A, B, C, D\}$, в виде генотипа $(1, 2, 3, 4)$. Такое представление может иметь место, например, для задачи коммивояжера, где A, B, C, D – вершины графа, а генотип $(1, 2, 3, 4)$ кодирует вершину A как 1, B как 2 и т. д., а порядок следования чисел в генотипе означает порядок обхода этих вершин. Если при этом размер фенотипа меняется, то меняется и размер генотипа. Подобного рода кодирование можно встретить в задачах, сводимых к поиску пути между двумя вершинами на графе. При этом число промежуточных узлов может меняться. Тогда генотип при прямом кодировании будет переменного размера, как в работе [17], где авторы используют прямое кодирование и предлагают свой вариант оператора скрещивания для генотипов (которые иногда называют хромосомами) переменной длины.

Прямое кодирование никак не решает проблемы роста ресурсоемкости с разрастанием вектора параметров целевой функции, а также не влияет на его переменную длину.

При непрямом кодировании (indirect encoding) уже не существует взаимно-однозначного соответствия между структурным элементом фенотипа и геном. Существует несколько различных способ реализации непрямого кодирования, например, использование L-систем.

Простейший случай L-систем можно описать тройкой (A, P, C), где A – алфавит (множество допустимых символов), P – правила вывода/замены/перезаписи (rewriting rules), C – аксиома (начальная строка из символов алфавита A). Все символы в строке C параллельно заменяются новыми символами в соответствии с правилами вывода P. В рамках одной итерации разные символы заменяются параллельно. Подвид грамматического кодирования, такой как L-системы, может иметь несколько разновидностей: параметрические L-системы и контекстно зависимые L-системы [18].

Как показывает обзор [18] все виды ранее упомянутой формы грамматического кодирования часто использовались для эволюции нейронных сетей (их топологии, матриц связности и т. д.), моделирования форм растений, а также для развития морфологий тела искусственных существ. Эффективное применение L-систем возможно лишь для эволюции таких объектов, которые могут быть представлены в виде рекурсивных выражений, пусть и с весьма сложными правилами перехода от одного выражения к другому. Последнее обстоятельство является ограничивающим фактором использования L-систем.

Отдельные компоненты L-системы можно закодировать в генотип и подвергнуть эволюции. Затем эволюционировавший генотип может использоваться для интерпретации L-системы, которая (интерпретация) используется как фенотип для целевой функции F. В работе [19] авторы используют генетический алгоритм для

эволюции правил перезаписи для L-систем, которые представляют собой 3D-объекты. Особенность применения ГА в данном контексте заключается в том, что в качестве функции оценки фенотипов используется ранжирование, осуществляемое пользователем на основе субъективной информации, так называемый «интерактивный генетический алгоритм». Затем полученные правила вывода при помощи ГА используются для порождения 3D-объектов, которые подлежат оценке.

Еще один из способов реализации непрямого кодирования заключается в использовании клеточного кодирования (cellular encoding). Клеточное кодирование является, как и L-системы, разновидностью грамматического кодирования. Клеточное кодирование в основном использовалось для построения нейронных сетей [18]. Клеточное кодирование традиционно применяется к графам и может быть формально представлено как тройка: (G, R, T). Где G – исходный граф, R – правила замены, T – маркировка вершин графа. Граф G зачастую состоит из одной вершины. Правила R применяются относительно всех вершин, имеющих определенную метку, которая задается T. Примером такого правила (элемента R) может быть разделение всех вершин с меткой А на две вершины с метками А и Б. Все правила замены применяются итеративно и параллельно в рамках каждой итерации.

Кодирование фенотипа на основе клеточного кодирования представлено в работе [20]. В ней применяют программирование экспрессии генов (Gene Expression Programming – GEP) к решению задачи поиска эффективных сверточных нейросетей (Convolutional Neural Network – CNN). Фенотипом выступает сверточная нейронная сеть, для кодирования которой авторы используют метод, основанный на клеточном кодировании. Ген в генотипе разделен на 2 части: заголовок и хвост. В заголовке содержится всегда одинаковое количество правил замены клеточного кодирования. В хвосте содержатся операции свертки, количество которых на единицу больше количества правил замены в заголовке. Элементы хвоста являются вершинами графа. В качестве правил замены используются три различных операции разделения вершины и еще одно правило останавливающее развитие графа. Эксперименты на наборах данных CIFAR-10 и CIFAR-100 показывают, что предложенный подход позволяет отыскивать эффективные архитектуры CNN.

Непрямое кодирование решает проблему разрастания вектора параметров целевой функции и может обеспечить его постоянный размер. Однако существующие методы, такие как L-системы и клеточное кодирование, сильно ограничены в своей выразительной мощности, что делает их пригодными лишь для небольшого класса специализированных задач. Они не применяются для кодирования фенотипа произвольной структуры.

В данной работе авторы ставят перед собой цель разработать метод преобразования вектора параметров целевой функции переменного размера в вектор параметров целевой функции постоянного размера, что позволит применять любые эволюционные методы оптимизации нулевого порядка к задачам SLAP, CSP и SSP.

Материалы и методы

L-системы и клеточное кодирование имеют одну общую особенность, которая наталкивает на их обобщение. Основная идея, которая используется в кодировании на основе L-систем заключается в том, что эволюционный алгоритм оперирует генотипом меньшего размера по сравнению с фенотипом. Более того, фенотип порождается при помощи некоторого алгоритма (развитие L-системы) на основе генотипа. Как правило, для кодирования на основе L-систем генотип представляет их правило перезаписи. Тоже самое можно сказать про клеточное кодирование: эволюционный алгоритм применяется

относительно генотипов, представляющих правила замены R. После эволюции, получившиеся правила применяются к исходному графу G с маркировкой T и получается по некоторому алгоритму фенотип.

Выделяя общее из этих двух способов и абстрагируясь от деталей реализации, приходим к следующему: пусть дана целевая функция $F: X \to Y$ (F является отображением X в Y), у которой элементы $x \in X$ могут иметь произвольную структуру (в том числе могут иметь различную размерность, если последняя для элементов множества X определена). Также пусть имеется функция $D: V \to X$. Тогда осуществим переход от задачи оптимизации (Рисунок 1а)

$$\min_{x \in X} F(x) \tag{1}$$

к задаче оптимизации (Рисунок 1б)

$$\min_{v \in V} F(D(v)). \tag{2}$$

Таким образом, при переходе от оптимизации функции F к оптимизации функции $F \circ D$ поисковое множество изменилось с X на V.

Для кодирования на основе L-систем множеством V является множество возможных параметров L-системы (в большинстве реализаций, V – множество возможных правил перезаписи, но для параметризованных или контекстно-зависимых L-систем может быть целый кортеж элементов). Если в качестве генотипов выбираются элементы множества V, то их размер будет практически всегда намного меньше размера генотипов, которые при прямом кодировании являлись бы развитыми L-системами (которые были бы фенотипами) из множества X. Функция D – алгоритм развития L-системы относительно элементов из множества V, например, если в качестве элементов V были правила перезаписи, то функция D развивает аксиому относительно заданного правила перезаписи $v \in V$, порождая фенотип $x \in X$. Затем целевая функция может быть вычислена относительно этого фенотипа.

Для клеточного кодирования ситуация аналогична. В качестве множества V выступает множество возможных параметров клеточного кодирования (зачастую V – множество возможных правил замены). Если в качестве генотипов выбираются элементы из V, то их размер практически всегда будет меньше размера генотипа, которым при прямом кодировании мог бы быть измененный граф из множества X. Функция D в данном случае, изменяла бы исходный граф на основе параметров v из множества V, например, если бы элементами V были правила замены, то D изменяла бы исходный граф в соответствии с правилами замены из V, порождая фенотип $x \in X$.

На самом деле данный метод обобщает и прямое кодирование: достаточно положить, чтобы в этом случае D была биекцией X на X, т. е. $D: X \leftrightarrow X$. В частном случае можно потребовать:

$$\forall x \in X (D(x) = x).$$

Таким образом, описанный метод можно использовать и для генетического алгоритма с генотипом постоянной длины.

Однако есть один нюанс. Переход от задачи (1) к задаче (2) приводит к тому, что функция F вычисляется теперь лишь от тех элементов, которые принадлежат образу $D(V)$. В зависимости от функции D этот факт может существенно сузить область поиска, что может привести к преждевременной сходимости даже для методов глобального поиска (вроде генетических алгоритмов). Таким образом, большое значение имеет выбор функции D: крайне важна ее область определения ($dom(D)$), которая

определяет размер генотипа, и множество значений ($rng(D)$), которое определяет область поиска.

Отдельного внимания заслуживает случай, при котором размер вектора параметров $v \in V$ фиксирован. Тогда является возможным использование всех имеющихся алгоритмов оптимизации соответствующего класса. Например, для задач комбинаторной оптимизации с вектором параметров постоянной длины применимы алгоритмы оптимизации нулевого порядка, в том числе алгоритмы с векторами параметров переменной длины, т. к. векторы постоянной длины являются их частным случаем.

Таким образом, предложенный формализованный метод обобщает прямое и непрямое кодирования, в частности, кодирование на основе L-систем и клеточное кодирование.

Данный метод можно повторно применить, но уже к функции D, т. е. перейти к оптимизации композиции трех функций, что даст переход от задачи (2) к задаче:

$$\min_{v \in V_1} F(D(D_1(v))),$$

где $D_1 : V_1 \to V$ (Рисунок 1в). Положим:

$$D_i : V_i \to V_{i-1}, i = 1 \ldots M,$$

тогда последовательное применение данного метода $M + 1$ раз даст переход от задачи (1) к задаче (Рисунок 1г):

$$\min_{v \in V_M} F \circ D \circ D_1 \ldots \circ D_M(v).$$

Использование композиции функций имеет практическое значение, поскольку позволяет управлять сложностью преобразования векторов параметров. Каждая функция может отвечать за отдельный алгоритм, отображающий один вектор параметров в другой вектор параметров.

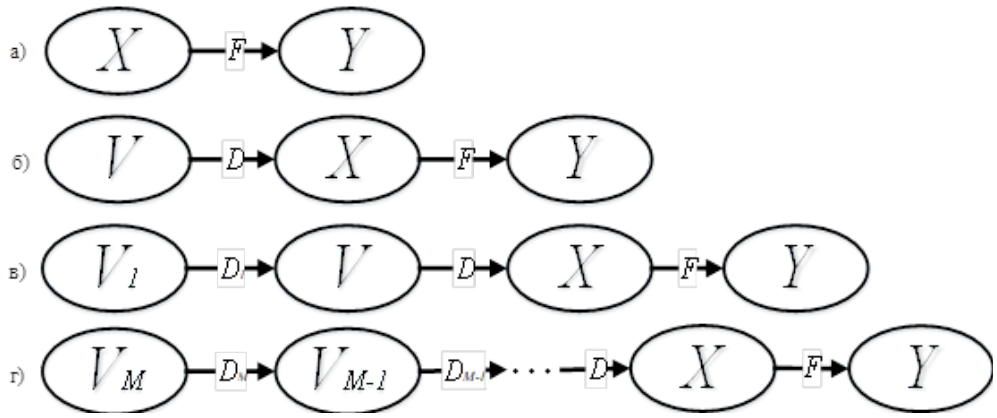

Рисунок 1 – Область определения и множество значений целевой функции: а – до применения метода перехода к вектору параметров постоянного размера; б – после применения метода перехода к вектору параметров постоянного размера; в – повторное применение метода перехода к вектору параметров постоянного размера; г – применение метода перехода к вектору параметров постоянного размера $M + 1$ раз

Figure 1 – The domain and range of the objective function: a – before applying the method of transitioning to a fixed-size parameter vector; b – after applying the method of transitioning to a fixed-size parameter vector; c – reapplying the method of transitioning to a fixed-size parameter vector;d – applying the method of transitioning to a fixed-size parameter vector $M + 1$ times

Также примем следующие обозначения: h_s – безопасная высота, на которую должен быть поднят сляб при транспортировке; h_m – максимальная допустимая высота штабелей (в слябах); K – количество слябов, которые необходимо извлечь и отправить на прокатку (количество элементов в графике горячей прокатки); $u(x)$ – позиция (считая снизу-вверх) сляба x в его штабеле; $r(x)$ – количество слябов в штабеле, в котором расположен сляб x; P_x – множество всевозможных положений крана по оси x ($P_x \subset \mathbb{R}$); y_x – множество всевозможных положений крана по оси y ($P_y \subset \mathbb{R}$); P_z – множество всевозможных положений крана по оси z ($P_z \subset \mathbb{R}$); S – множество всех слябов на складе. Более формально и подробно данные обозначения и математическая модель склада слябов разобраны в работе [2].

Множествами всевозможных атомарных команд будут:

$$O_1 = \{o_1(c) | c \in P_x\}, O_2 = \{o_1(c) | c \in P_y\},$$

$$O_3 = \{o_3(c) | c \in P_z\}, O_4 = \{o_4(c) | c \in P_z\},$$

$$O_7 = \{o_7(c) | c \in \mathbb{R} \wedge c > 0\},$$

а множеством всех возможных атомарных команд будет:

$$O_{all} = O_1 \cup O_2 \cup O_3 \cup O_4 \cup O_7 \cup \{o_5\} \cup \{o_6\}.$$

Оптимизация совмещенных CSP и SSP обычно преследует цель минимизации пройденного расстояния кранами, балансировку нагрузки между кранами и т. п. Пусть

$$X = \bigcup_{i=1}^{\infty} {O_{all}}^i$$

и $Y = \mathbb{R}$, тогда целевая функция $F: X \to Y$ оценивает составленное расписание кранов для заданного графика горячей прокатки (например, ставит в соответствие расписанию крана пройденное им расстояние). Легко видеть, что множество $dom(F)$ состоит из векторов различной длины. Используя метод перехода к вектору параметров постоянного размера к функции F, можно добиться перехода от множества последовательностей атомарных команд переменной длины к последовательности команд постоянной длины.

Для этого определим следующие виртуальные команды первого уровня: $v_1(x, y, z)$ – подъем мостовым краном сляба с позиции (x, y, z); $v_2(x, y, z)$ – опускание захваченного сляба в позицию (x, y, z).

Каждая виртуальная команда первого уровня отображается в последовательность фиксированного размера атомарных команд следующим образом:

$$v_1(x, y, z) \to o_1(x), o_2(y), o_3(h_s - z), o_5, o_4(h_s - z),$$

$$v_2(x, y, z) \to o_1(x), o_2(y), o_3(h_s - z), o_6, o_4(h_s - z).$$

Также определим виртуальную команду второго уровня, которая будет транслироваться во множество виртуальных команд первого уровня: $v'(s)$ – отправить сляб s на горячую прокатку.

Количество виртуальных команд первого уровня, в которые должна транслироваться виртуальная команда второго уровня, зависит от извлекаемого сляба. Определим множества всех виртуальных команд первого уровня:

$$V'_1 = \{v_1(x, y, z) | x \in P_x, y \in P_y, z \in P_z\},$$

$$V'_2 = \{v_2(x, y, z) | x \in P_x, y \in P_y, z \in P_z\}.$$

На их основе определим множество всех виртуальных команд первого уровня:

$$V'_{all} = V'_1 \cup V'_2.$$

Множество всех возможных виртуальных команд второго уровня определяется аналогично:

$$V_A = \{v'_1(s) | s \in S\}.$$

Пусть $V = \bigcup_{i=1}^{\infty} V'_{all}{}^{i}$ и $V_1 = V_A{}^{K}$. Тогда $D: V \to X$ транслирует любую заданную последовательность виртуальных команд первого уровня в последовательность атомарных команд, применяя описанное выше преобразование к каждому элементу вектора $v \in V$. Функция $D_1: V_1 \to V$ выполняет преобразование последовательности виртуальных команд второго уровня в последовательность виртуальных команд первого уровня, применяя алгоритм, показанный на Рисунке 2, к каждому элементу вектора $v \in V_1$ и объединяя в один кортеж результирующие последовательности команд. Элементы $v \in V_1$ задают последовательность команд на извлечение и отправку на прокатку K слябов.

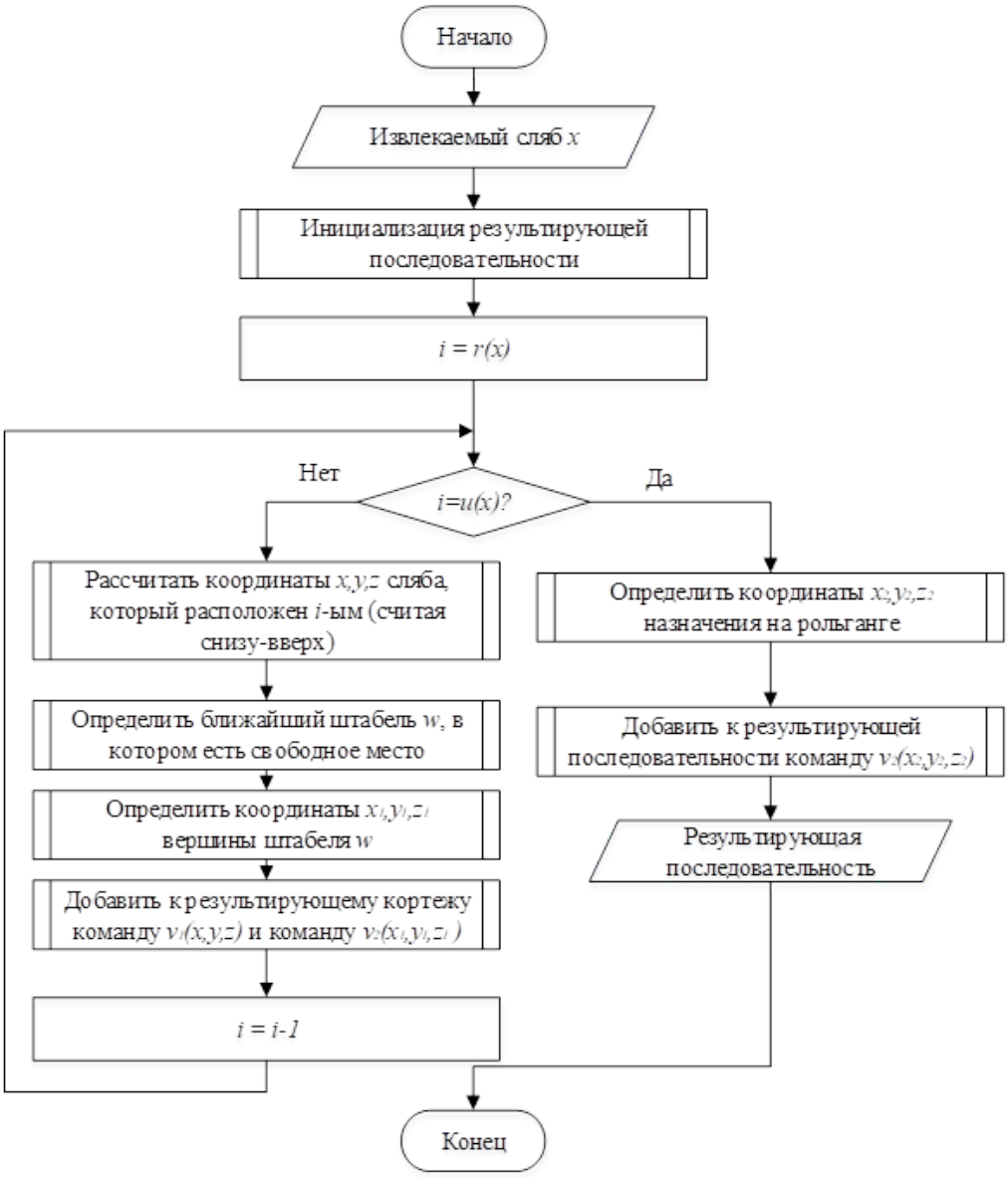

Рисунок 2 – Алгоритм трансляции виртуальной команды второго уровня
Figure 2 – Algorithm for translating a second-level virtual command

Таким образом, для отправки на прокатку K слябов $s_1, s_2, \ldots, s_K$ в заданной последовательности, составим вектор виртуальных команд второго уровня:

$$\left(v'(s_1), v'(s_2), \ldots, v'(s_N)\right) \in V_1.$$

Затем функция D_1 транслирует каждую команду v' в последовательность команд v_1 и v_2 и объединяет их в кортеж, т. е. отображает вектор из V_1 в вектор из V. После этого функция D транслирует каждую виртуальную команду первого уровня в последовательность атомарных команд, а затем объединяет их в кортеж, т. е. отображает вектор из V в вектор из X. Наконец, целевая функция F выполнит оценку этого вектора, возвращая вещественное число из $\mathbb{R}$. Данная ситуация продемонстрирована на Рисунке 1в.

Стоит отметить, что для заданного графика горячей прокатки векторы из V_1 будут все одинаковой длины (равной K), что позволяет применять любые известные методы оптимизации нулевого порядка к композиции трех функций $F \circ D \circ D_1$. Наименьшая длина вектора $v \in V$ равняется $2K$, т. к. для извлечения каждого из K слябов требуется минимум две виртуальные команды первого уровня: v_1 и v_2 – и наименьшее количество виртуальных команд первого уровня потребуется, если каждый раз мостовой кран извлекает самый верхний сляб в штабеле. Т. к. обе виртуальные команды первого уровня транслируются в 5 атомарных команд, то наименьшая длина вектора $x \in X$ будет равняться $10\,K$. Таким образом, длина вектора параметров целевой функции, после применения предложенного авторами метода, становится как минимум в 10 раз меньше и имеет постоянный размер по сравнению с вектором $x \in X$, который мог бы быть генотипом при прямом кодировании.

Результаты

Пусть при решении задачи SSP получилась следующая последовательность слябов (слябы однозначно определяются их идентификатором, который является целым числом) для графика горячей прокатки из 3-х элементов: 22, 6, 14, как показано на Рисунке 3а (для удобства штабели отображены в двумерном пространстве, а снизу подписаны координаты их центров). Высота всех слябов равна 20 сантиметрам. Максимальная высота штабеля равна 5 слябам. На Рисунке 3б показано состояние склада после отправки на прокатку сляба 22. На Рисунке 3в отображено расположение слябов после отправки на прокатку сляба 6. На Рисунке 3г показано состояние склада после отправки на прокатку сляба 14. Сплошной заливкой показаны слябы, которые были переложены, а заштрихованы те слябы, которые требуется извлечь.

Для заданной последовательности извлечения слябов вектор $v \in V_1$ будет иметь вид:

$$(v'(22), v'(6), v'(14)).$$

На Рисунке 4а,в,д показана трансляция при помощи алгоритма виртуальных команд второго уровня, приведенного на Рисунке 3. Для команды $v'(22)$ число виртуальных команд первого уровня, в которые она транслируется, равно 4. Аналогичные значения для команд $v'(6)$ и $v'(14)$ равны 6 и 8 соответственно.

На Рисунке 4б,г,е показана трансляция виртуальных команд первого уровня во множество атомарных команд. Каждая виртуальная команда первого уровня транслируется в 5 атомарных команд. Предполагается, что слябы переносятся на конвейер, высота которого равна 20 сантиметров, а по оси y его координаты равны -40.

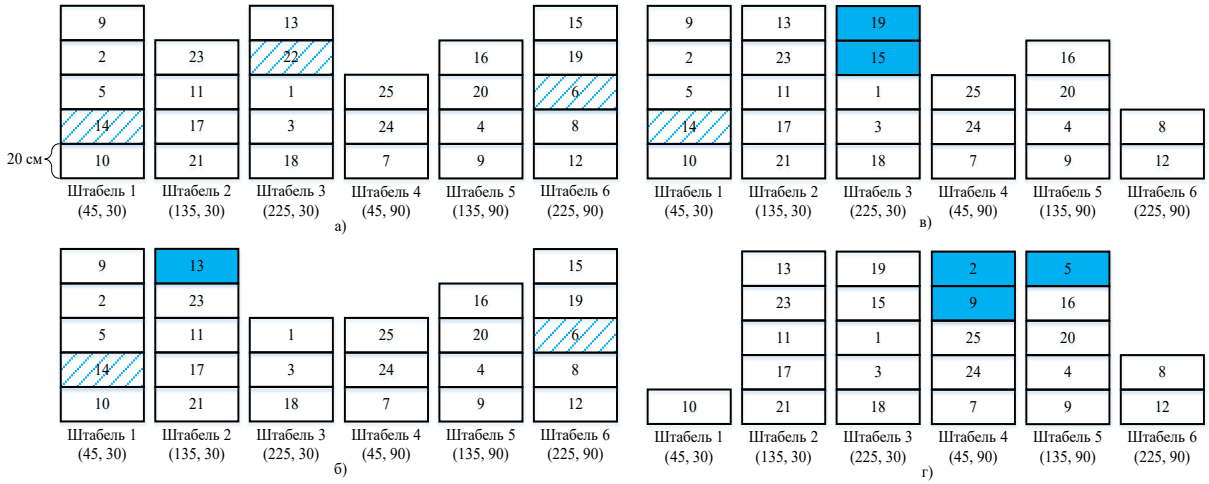

Рисунок 3 – Состояние склада слябов: а – изначальное расположение слябов; б – расположение слябов после отправки на прокатку сляба 22; в – расположение слябов после отправки на прокатку сляба 6; г – расположение слябов после отправки на прокатку сляба 14

Figure 3 – Slab yard state: a – initial slab layout; b – slab layout after slab 22 is sent for rolling; c – slab layout after slab 6 is sent for rolling; d – slab layout after slab 14 is sent for rolling

$$v'(22) \longrightarrow \begin{matrix} v_1(225,30,90), v_2(135,30,90), \\ v_1(225,30,70), v_2(225,-40,50) \end{matrix}$$
а)

$$\begin{aligned} v_1(225,30,90) &\longrightarrow o_1(225), o_2(30), o_3(h_s-90), o_5, o_4(h_s-90) \\ v_2(135,30,90) &\longrightarrow o_1(135), o_2(30), o_3(h_s-90), o_6, o_4(h_s-90) \\ v_1(225,30,70) &\longrightarrow o_1(225), o_2(30), o_3(h_s-70), o_5, o_4(h_s-70) \\ v_2(225,-40,50) &\longrightarrow o_1(225), o_2(-40), o_3(h_s-50), o_6, o_4(h_s-50) \end{aligned}$$
б)

$$v'(6) \longrightarrow \begin{matrix} v_1(225,90,110), v_2(225,30,70) \\ v_1(225,90,70), v_2(225,30,90) \\ v_1(225,90,50), v_2(225,-40,50) \end{matrix}$$
в)

$$\begin{aligned} v_1(225,90,110) &\longrightarrow o_1(225), o_2(90), o_3(h_s-110), o_5, o_4(h_s-110) \\ v_2(225,30,70) &\longrightarrow o_1(225), o_2(30), o_3(h_s-70), o_6, o_4(h_s-70) \\ v_1(225,90,70) &\longrightarrow o_1(225), o_2(90), o_3(h_s-70), o_5, o_4(h_s-70) \\ v_2(225,30,90) &\longrightarrow o_1(225), o_2(30), o_3(h_s-90), o_6, o_4(h_s-90) \\ v_1(225,90,50) &\longrightarrow o_1(225), o_2(90), o_3(h_s-50), o_5, o_4(h_s-50) \\ v_2(225,-40,50) &\longrightarrow o_1(225), o_2(-40), o_3(h_s-50), o_6, o_4(h_s-50) \end{aligned}$$
г)

$$v'(14) \longrightarrow \begin{matrix} v_1(45,30,90), v_2(45,90,70), \\ v_1(45,30,70), v_2(45,90,90), \\ v_1(45,30,50), v_2(135,90,90), \\ v_1(45,30,30), v_2(45,-40,50) \end{matrix}$$
д)

$$\begin{aligned} v_1(45,30,90) &\longrightarrow o_1(45), o_2(30), o_3(h_s-90), o_5, o_4(h_s-90) \\ v_2(45,90,70) &\longrightarrow o_1(45), o_2(90), o_3(h_s-70), o_6, o_4(h_s-70) \\ v_1(45,30,70) &\longrightarrow o_1(45), o_2(30), o_3(h_s-70), o_5, o_4(h_s-70) \\ v_2(45,90,90) &\longrightarrow o_1(45), o_2(90), o_3(h_s-90), o_6, o_4(h_s-90) \\ v_1(45,30,50) &\longrightarrow o_1(45), o_2(30), o_3(h_s-50), o_5, o_4(h_s-50) \\ v_2(135,90,90) &\longrightarrow o_1(135), o_2(90), o_3(h_s-90), o_6, o_4(h_s-90) \\ v_1(45,30,30) &\longrightarrow o_1(45), o_2(30), o_3(h_s-30), o_5, o_4(h_s-30) \\ v_2(45,-40,50) &\longrightarrow o_1(45), o_2(-40), o_3(h_s-50), o_6, o_4(h_s-50) \end{aligned}$$
е)

Рисунок 4 – Трансляция команд: а – трансляция виртуальной команды $v'(22)$; б – трансляция виртуальных команд первого уровня, полученных при трансляции команды $v'(22)$, в атомарные команды; в – трансляция виртуальной команды $v'(6)$; г – трансляция виртуальных команд первого уровня, полученных при трансляции команды $v'(6)$, в атомарные команды; д – трансляция виртуальной команды $v'(14)$; е – трансляция виртуальных команд первого уровня, полученных при трансляции команды $v'(14)$, в атомарные команды

Figure 4 – Commands translation: a – translation of the virtual command $v'(22)$; b – translation of the first-level virtual commands obtained during the translation of command $v'(22)$ into atomic commands; c – translation of the virtual command $v'(6)$; d – translation of first-level virtual commands received during the translation of command $v'(6)$ into atomic commands; e – translation of virtual command $v'(14)$; f – translation of first-level virtual commands received during the translation of command $v'(14)$ into atomic commands

На Рисунке 5 показаны отображения: вектора виртуальных команд второго уровня при помощи функции D_1 в вектор виртуальных команд первого уровня; вектора виртуальных команд первого уровня при помощи функции D в вектор атомарных команд; вектора атомарных команд при помощи функции F в вещественное число x.

Можно видеть, что вектор из поискового множества V_1 длины 3 отображается при помощи композиции $D \circ D_1$ в вектор атомарных команд длины 45.

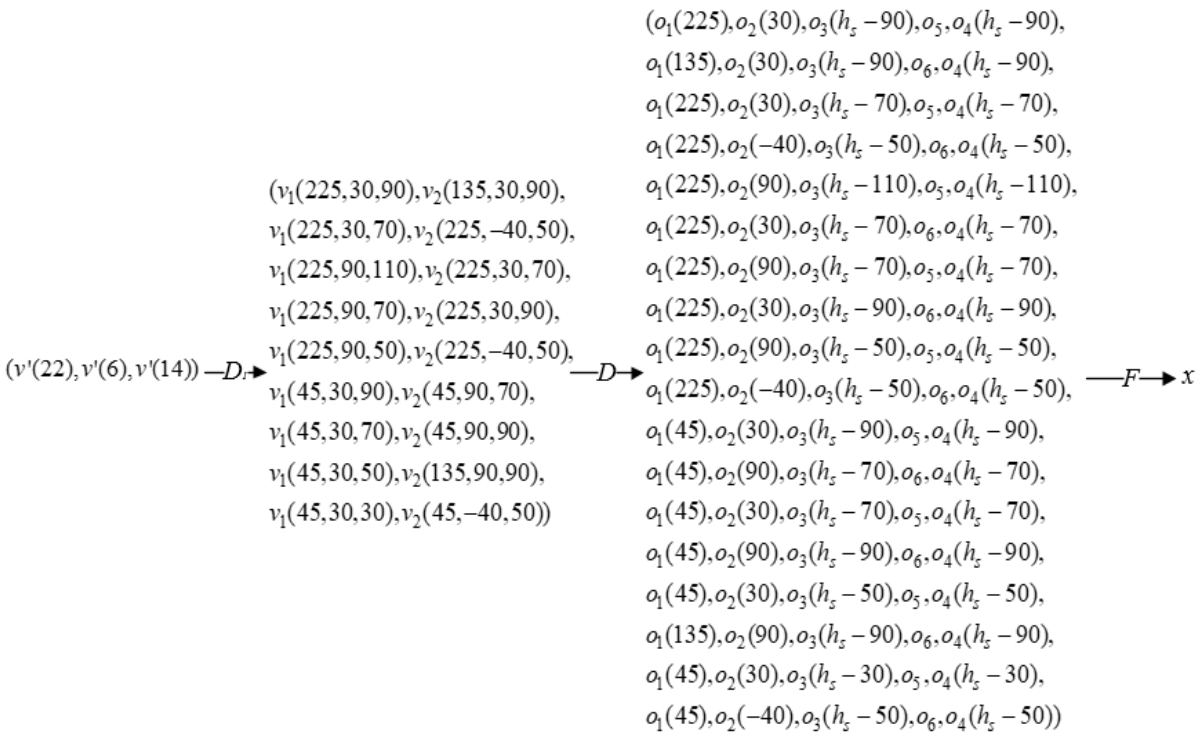

Рисунок 5 – Отображения векторов команд при помощи функций D_1, D и F
Figure 5 – Mapping of command vectors using the D_1, D and F functions

Стоит отметить, что аналогичная схема кодирования применяется в работе [21]. В данной работе авторы определили целевую функцию таким образом, что она принимает на вход последовательность номеров извлекаемых слябов, а на выходе дает оценку этого вектора в виде количества требуемых перекладываний слябов. Эта целевая функция интерпретирует вектор параметров, для каждого из которых она определяет количество слябов необходимых для перекладывания. Решения, полученные при помощи генетического алгоритма, оказались более эффективными (требовали меньшее количество перекладываний для заданного графика прокатки) по сравнению с применяемой на складе слябов жадной эвристикой, что подтверждает сходимость генетических алгоритмов в задачах подобного рода с описанным кодированием. Однако в [21] подобный подход использовался как частный эвристический способ. Предложенный нами метод позволит исследователям явным образом переходить к новому поисковому пространству в комбинаторных задачах, таких как SSP и CSP.

Обсуждение

Применение предложенного авторами метода к задачам составления расписания мостовых кранов позволяет построить гибкую иерархию команд. Команды разных уровней обладают разной степенью детализации. Абстрактные, виртуальные команды второго уровня могут задавать требуемую операцию над слябом (что может быть

полезно специалистам логистического центра). Виртуальные команды первого уровня могут использоваться для логического управления мостовыми кранами, а атомарные команды в свою очередь нужны для фактического управления отдельными элементами этих кранов.

Данную иерархию команд можно использовать для создания цифрового двойника склада слябов. На вход такому двойнику будут подаваться виртуальные команды второго уровня, а в качестве результирующего значения цифровой двойник вернет расписание для мостовых кранов. Это, в свою очередь, поможет перейти к полностью автоматизированному складу непрерывнолитых заготовок, что поможет снизить износ оборудования, повысить эффективность его использования и снизить риски для персонала.

С теоретической точки зрения применение данного метода не ограничивается задачами оптимизации составления расписания мостовых кранов. Возможно дальнейшее исследование предложенного метода для оптимизационных задач первого и высших порядков.

Заключение

В данной работе проанализирован методологический аппарат на основе эволюционных вычислений для кодирования вектора параметров переменной длины. В частности, рассмотрены: прямое кодирование, L-системы, клеточное кодирование и непрямое кодирование.

На основе проанализированного материала был предложен метод, позволяющий преобразовать векторы параметров целевой функции переменной длины к векторам параметров постоянной длины. Этот переход осуществляется за счет вспомогательных функций, в результате чего меняется и поисковое множество целевой функции. Предложенный метод позволяет применить все многообразие методологии оптимизации (в том числе не из области эволюционных вычислений) для решения исходной задачи.

Конкретное применение метода перехода к вектору параметров постоянной длины продемонстрировано на примере целевой функции, которая возникает при совместном решении оптимизационных задач CSP и SSP. Рассматриваемый метод сжимает произвольный вектор параметров целевой функции для этих задач минимум в 10 раз. Дальнейшие варианты развития настоящей работы могут заключаться в исследовании влияния применения описанного авторами метода на сходимость алгоритмов, реализующих методы оптимизации нулевого, первого и более высоких порядков, а также в применении предложенного метода к проектированию и реализации цифрового двойника склада слябов.

СПИСОК ИСТОЧНИКОВ / REFERENCES

1. Щеголихин И.С., Андреев С.М. Математическая модель расписания кранов для графика горячей прокатки непрерывнолитых заготовок на складе слябов. *Автоматизированные технологии и производства*. 2024;(1):13–16.
Shchegolikhin I.S., Andreev S.M. Mathematical model of crane scheduling for the schedule of hot rolling of continuous cast billets in the slab warehouse. *Automation of Technologies and Production*. 2024;(1):13–16. (In Russ.).
2. Shchegolikhin I.S., Andreev S.M. Mathematical Model of Crane Scheduling for a Defined Slab Storage Locations Assignment in a Continuous Cast Billet Warehouse. In: *2024 International Russian Automation Conference (RusAutoCon), 08–14 September 2024, Sochi, Russia*. IEEE; 2024. P. 1193–1199. https://doi.org/10.1109/RusAutoCon61949.2024.10694388

3. Kuyama S., Tomiyama S. A Two-phase Heuristic for Crane Scheduling in Steel Slab Yards. *IFAC Proceedings Volumes*. 2013;46(24):388–393. https://doi.org/10.3182/20130911-3-BR-3021.00039
4. Bruno G., Cavola M., Diglio A., et al. A unifying framework and a mathematical model for the Slab Stack Shuffling Problem. *International Journal of Industrial Engineering Computations*. 2023;14(1):17–32. https://doi.org/10.5267/j.ijiec.2022.10.005
5. Peng G., Wu Y., Zhang Ch., et al. Integrated optimization of storage location assignment and crane scheduling in an unmanned slab yard. *Computers & Industrial Engineering*. 2021;161(3):107623. https://doi.org/10.1016/j.cie.2021.107623
6. Tang L., Liu J., Rong A., et al. An effective heuristic algorithm to minimise stack shuffles in selecting steel slabs from the slab yard for heating and rolling. *Journal of the Operational Research Society*. 2001;52(10):1091–1097. https://doi.org/10.1057/palgrave.jors.2601143
7. Lu Ch., Zhang R., Liu Sh. A 0-1 integer programming model and solving strategies for the slab storage problem. *International Journal of Production Research*. 2016;54(8):2366–2376. https://doi.org/10.1080/00207543.2015.1076949
8. Shchegolikhin I.S., Andreev S.M. Development and Analysis of Genetic Algorithm for Optimization of Continuous Cast Billets Warehousing Process. In: *2024 International Russian Smart Industry Conference (SmartIndustryCon), 25–29 March 2024, Sochi, Russia*. IEEE; 2024. P. 962–967. https://doi.org/10.1109/SmartIndustryCon61328.2024.10515628
9. Erwin K., Engelbrecht A. Meta-heuristics for portfolio optimization. *Soft Computing*. 2023;27(24):19045–19073. https://doi.org/10.1007/s00500-023-08177-x
10. Hwang Sh.-F., He R.-S. A hybrid real-parameter genetic algorithm for function optimization. *Advanced Engineering Informatics*. 2006;20(1):7–21. https://doi.org/10.1016/j.aei.2005.09.001
11. Ono I., Kita H., Kobayashi Sh. A Real-coded Genetic Algorithm using the Unimodal Normal Distribution Crossover. In: *Advances in Evolutionary Computing: Theory and Applications*. Berlin, Heidelberg: Springer; 2003. P. 213–237. https://doi.org/10.1007/978-3-642-18965-4_8
12. Herrera F., Lozano M. Gradual distributed real-coded genetic algorithms. *IEEE Transactions on Evolutionary Computation*. 2000;4(1):43–63. https://doi.org/10.1109/4235.843494
13. Fontes D.B.M.M., Homayouni S.M., Gonçalves J.F. A hybrid particle swarm optimization and simulated annealing algorithm for the job shop scheduling problem with transport resources. *European Journal of Operational Research*. 2022;306(3):1140–1157. https://doi.org/10.1016/j.ejor.2022.09.006
14. Wang J., Liu Ch., Li K. A hybrid simulated annealing for scheduling in dual-resource cellular manufacturing system considering worker movement. *Automatika*. 2019;60(2):172–180. https://doi.org/10.1080/00051144.2019.1603264
15. Riazi A. Genetic algorithm and a double-chromosome implementation to the traveling salesman problem. *SN Applied Sciences*. 2019;1(11):1397. https://doi.org/10.1007/s42452-019-1469-1
16. Yang J., Wu Ch., Lee H.P., et al. Solving traveling salesman problems using generalized chromosome genetic algorithm. *Progress in Natural Science*. 2008;18(7):887–892. https://doi.org/10.1016/j.pnsc.2008.01.030
17. Zhang Q., Ding L. A new crossover mechanism for genetic algorithms with variable-length chromosomes for path optimization problems. *Expert Systems with Applications*. 2016;60:183–189. https://doi.org/10.1016/j.eswa.2016.04.005

18. Stanley K.O., Miikkulainen R. A Taxonomy for Artificial Embryogeny. *Artificial Life*. 2003;9(2):93–130. https://doi.org/10.1162/106454603322221487
19. Ariffin M.K.b., Hadi Sh., Phon-Amnuaisuk S. Evolving 3D Models Using Interactive Genetic Algorithms and L-Systems. In: *Multi-disciplinary Trends in Artificial Intelligence: 11th International Workshop, 20–22 November 2017, Gadong, Brunei*. Cham: Springer; 2017. P. 485–493. https://doi.org/10.1007/978-3-319-69456-6_40
20. Broni-Bediako C., Murata Y., Mormille L.H.B., et al. Evolutionary NAS with Gene Expression Programming of Cellular Encoding. In: *2020 IEEE Symposium Series on Computational Intelligence (SSCI), 01–04 December 2020, Canberra, Australia*. IEEE; 2020. P. 2670–2676. https://doi.org/10.1109/SSCI47803.2020.9308346
21. Tang L., Liu J., Rong A., et al. Modelling and a genetic algorithm solution for the slab stack shuffling problem when implementing steel rolling schedules. *International Journal of Production Research*. 2002;40(7):1583–1595. https://doi.org/10.1080/00207540110110118424

ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT THE AUTHORS

Щеголихин Иван Сергеевич, аспирант, ассистент Магнитогорского государственного технического университета им. Г.И. Носова, Магнитогорск, Российская Федерация.
e-mail: shchegolikhin.i@yandex.ru
ORCID: 0009-0008-2239-983X

Ivan S. Shchegolikhin, Postgraduate, Assistant Professor of G.I. Nosov Magnitogorsk State Technical University, Magnitogorsk, the Russian Federation.

Андреев Сергей Михайлович, доктор технических наук, доцент Магнитогорского государственного технического университета им. Г.И. Носова, Магнитогорск, Российская Федерация.
e-mail: andreev.asc@gmail.com
ORCID: 0000-0003-0735-6723

Andreev S. Mikhailovich, Doctor of Engineering Sciences, Associate Professor of G.I. Nosov Magnitogorsk State Technical University, Magnitogorsk, the Russian Federation.

Статья поступила в редакцию 29.05.2026; одобрена после рецензирования 27.07.2026; принята к публикации 07.08.2026.

The article was submitted 29.05.2026; approved after reviewing 27.07.2026; accepted for publication 07.08.2026.