

УДК 519.87

ЛОКАЛЬНЫЙ ПОИСК С ЧЕРЕДУЮЩИМИСЯ ОКРЕСТНОСТЯМИ ДЛЯ ДВУХСТАДИЙНОЙ ЗАДАЧИ РАЗМЕЩЕНИЯ

Т. В. Леванова, А. С. Федоренко

Аннотация. Предложены варианты алгоритма локального поиска с чередующимися окрестностями (VNS) для решения двухстадийной задачи размещения. В рамках данной схемы определены наборы окрестностей и порядок их просмотра для решения этой задачи. Введена новая рандомизированная окрестность, учитывающая специфику задачи и улучшающая работу алгоритма. Приведены результаты экспериментальных исследований предложенных алгоритмов на трудных в вычислительном отношении сериях тестовых задач. Показано, что использование новой окрестности существенно повышает качество работы алгоритма и позволяет находить решения с малой погрешностью за относительно небольшое время. Выполнено сравнение VNS с алгоритмом имитации отжига (SA), имеющим свойство асимптотической сходимости. Сравнительный анализ показал, что предложенный алгоритм превосходит соответствующие результаты SA на большинстве тестовых примеров.

Ключевые слова: дискретная оптимизация, двухстадийная задача размещения предприятий, локальный поиск с чередующимися окрестностями.

Введение

Многие прикладные задачи, возникающие при планировании и развитии производства, в стандартизации и других областях, сводятся к решению дискретных задач оптимального размещения. Среди исследований по оптимизации размещения предприятий важное место занимает двухстадийная задача, в которой продукция проходит две стадии производства [1]. Её частным случаем является простейшая задача размещения, которая NP-трудна в сильном смысле [13].

В настоящее время для решения многих дискретных оптимизационных задач активно используются эвристические методы [7, 9]. Актуальность их разработки и исследования связана, в первую очередь, с вычислительной сложностью и большой размерностью рассматриваемых

задач. Значительное внимание уделяется алгоритмам локального поиска с чередующимися окрестностями (VNS) для решения задач дискретной оптимизации, в том числе для задач оптимального размещения предприятий [6, 11, 17].

В данной статье предложены варианты алгоритмов VNS для двухстадийной задачи размещения [1], проведены экспериментальные исследования. В частности, выполнен анализ работы алгоритмов при различных настройках параметров с точки зрения точности получаемых решений, времени счета и др.

Статья организована следующим образом. В разд. 1 дается постановка двухстадийной задачи размещения. В разд. 2 приводится общая схема алгоритма локального поиска с чередующимися окрестностями для задач дискретной оптимизации. В разд. 3 предлагаются варианты VNS для двухстадийной задачи размещения. В разд. 4 содержатся результаты вычислительного эксперимента.

1. Постановка двухстадийной задачи размещения

В двухстадийной задаче размещения [1] заданы предприятия, выпускающие некоторую продукцию, и потребители этой продукции. Предприятия делятся на два множества, отражающие стадии производства продукции (нижний уровень) и ее реализации (верхний уровень), причем каждое предприятие верхнего уровня связано с несколькими предприятиями нижнего уровня. В удовлетворении спроса потребителей участвуют лишь предприятия верхнего уровня. Предполагается, что они могут обслуживать любого потребителя, который, в свою очередь, должен обслуживаться лишь одним предприятием. Предприятие верхнего уровня считается открытым, если таковыми являются все связанные с ним предприятия нижнего уровня. Заданы стоимости открытия предприятий, известны затраты на удовлетворение спроса каждого потребителя предприятиями верхнего уровня. Требуется найти набор предприятий, суммарные затраты на открытие которых и удовлетворение ими спроса всех потребителей минимальны.

Введем следующие обозначения:

$I = \{1, \dots, m\}$ — множество номеров предприятий верхнего уровня;

$L = \{1, \dots, k\}$ — множество номеров предприятий нижнего уровня;

$J = \{1, \dots, n\}$ — множество номеров потребителей;

$d_i \geq 0$ — стоимость открытия предприятия i верхнего уровня, $i \in I$;

$c_l \geq 0$ — стоимость открытия предприятия l нижнего уровня, $l \in L$;

g_{il} — величина, показывающая, связано или нет предприятие i верхнего

уровня с предприятием l нижнего уровня: $g_{il} = 1$, если предприятия i и l связаны, в противном случае $g_{il} = 0$, $i \in I$, $l \in L$;

t_{ij} — стоимость обслуживания потребителя j предприятием i , $i \in I$, $j \in J$.

Введем булевы переменные: $z_i = 1$, если предприятие i верхнего уровня открыто, иначе $z_i = 0$, $i \in I$. Аналогично определяются булевы переменные y_l для предприятий нижнего уровня, $l \in L$. Кроме того, $x_{ij} = 1$, если предприятие i удовлетворяет спрос потребителя j , $x_{ij} = 0$ в противном случае, $i \in I$, $j \in J$.

С использованием введенных обозначений математическая модель двухстадийной задачи размещения может быть записана следующим образом:

$$F(z, y, X) = \sum_{l \in L} c_l y_l + \sum_{i \in I} d_i z_i + \sum_{i \in I} \sum_{j \in J} t_{ij} x_{ij} \rightarrow \min, \quad (1)$$

$$\sum_{i \in I} x_{ij} = 1, \quad j \in J, \quad (2)$$

$$z_i \geq x_{ij}, \quad i \in I, j \in J, \quad (3)$$

$$y_l \geq g_{il} z_i, \quad i \in I, l \in L, \quad (4)$$

$$x_{ij}, y_l, z_i \in \{0, 1\}, \quad i \in I, j \in J, l \in L. \quad (5)$$

Целевая функция (1) отражает суммарные затраты на открытие предприятий верхнего и нижнего уровней и удовлетворение спроса потребителей. Равенства (2) гарантируют удовлетворение спроса каждого потребителя. Условия (3) отвечают за то, чтобы потребителей обслуживали только открытые предприятия верхнего уровня. Неравенства (4) показывают, что если предприятия верхнего уровня открыты, то также должны быть открыты и связанные с ними предприятия нижнего уровня.

Заметим, что в случае, когда все затраты на открытие предприятий нижнего уровня равны нулю, т. е. $c_l = 0$, $l \in L$, задача (1)–(5) превращается в простейшую задачу размещения (ПЗР), которой посвящено большое число исследований (например, [1, 2, 4]).

2. Локальный поиск с чередующимися окрестностями для задач дискретной оптимизации

В алгоритмах локального поиска важное место занимает понятие окрестности. Под окрестностью N вектора b будем понимать множество

векторов $N(b)$ в некотором смысле «близких» к данному. От определения «близости» зависит вид окрестности. Например, мерой близости для булевых векторов может служить расстояние Хемминга.

Стандартный алгоритм локального поиска [5] начинает работу с некоторого начального допустимого решения задачи. В процессе поиска происходит переход по заданной окрестности от текущего решения к новому с меньшим значением целевой функции, которое становится текущим. Этот процесс продолжается до тех пор, пока не будет найден локальный оптимум. Экспериментальные исследования показали, что для многих задач такой алгоритм быстро достигает локального оптимума, а найденные решения часто имеют большую погрешность. В связи с этим получили развитие методы, позволяющие выходить из локальных оптимумов [7, 9, 11], одним из которых является поиск с чередующимися окрестностями.

Идея алгоритма локального поиска с чередующимися окрестностями (Variable Neighborhood Search, VNS) основывается на следующих замечаниях: а) локальный оптимум относительно одной окрестности может не быть им относительно другой окрестности; б) глобальный оптимум является локальным оптимумом относительно любой окрестности. Первые работы по указанным алгоритмам выполнены Н. Младеновичем и П. Хансеном [16].

Таким образом, если в процессе работы алгоритма использовать различные окрестности, появляется возможность продолжить поиск лучшего решения, переходя от локального оптимума в одной окрестности к локальному оптимуму относительно другой окрестности.

Общая схема алгоритма локального поиска с чередующимися окрестностями состоит в следующем. В начале работы алгоритма определяем множество используемых окрестностей, порядок их просмотра и выбираем начальное решение. Начиная с первой окрестности, алгоритм за один шаг специальным образом просматривает список окрестностей и находит улучшающее решение. Поиск нового решения происходит по следующему правилу. Начиная с текущей точки, ищем локальный оптимум по отношению к первой окрестности из списка. Если это решение не улучшает значения целевой функции, для этого локального оптимума строим следующую окрестность и ищем в ней решение, улучшающее значение локального оптимума. Как только такое решение найдено, переходим на следующий шаг. Если в очередной окрестности улучшающее решение не найдено и список окрестностей не пройден, переходим к следующей окрестности текущего решения. Если весь список просмотрен и улуч-

шающее решение не найдено, то алгоритм останавливается, результатом работы является лучшее из найденных решений.

Ранее алгоритм локального поиска с чередующимися окрестностями был успешно применен для задачи коммивояжера, квадратичной задачи о назначениях, задачи о p -медиане, простейшей задачи размещения и ряда других [11]. Эти достижения привели к дальнейшим активным разработкам VNS для решения различных задач [17].

3. Алгоритм решения двухстадийной задачи размещения

В процессе разработки алгоритма локального поиска с чередующимися окрестностями необходимо было определить, какие окрестности и для каких векторов будут использоваться, их порядок, а также какая часть окрестностей будет просматриваться. С одной стороны, их сочетание должно позволять переходить от одного локального минимума к другому. С другой стороны, просмотр окрестностей должен занимать относительно небольшое время. Расположение локальных минимумов, сложность нахождения решения зависят от специфики задачи.

В данной работе использовались следующие известные виды окрестностей [5]: k -Flip, k -Swap, окрестность Лина — Кернигана (LK). Кроме того, была предложена окрестность «взвешенных решений».

Окрестность k -Flip булева вектора b содержит все точки b' , для которых расстояние Хемминга от вектора b не больше чем k .

Окрестность k -Swap булева вектора b содержит все точки b' , у которых ровно k единиц заменили на k нулей, а k нулей — на k единиц. В этом случае расстояние Хемминга от данного вектора b равно $2k$, и количество ненулевых компонент вектора не изменяется.

Окрестностью Лина — Кернигана называется множество точек, полученных следующим образом. В некоторой окрестности N данной точки b находится лучшее решение b^1 , которое считается первой точкой окрестности LK. Далее уже из окрестности $N(b^1)$ решения b^1 из множества $N(b^1) \setminus b$ выбирается лучшая точка b^2 , которая считается второй точкой окрестности. Процесс продолжается до тех пор, пока не получится требуемое количество точек в окрестности.

При построении окрестности «взвешенных решений» не более $3q$ компонент вектора b меняются, остальные нет. Для этого каждому предприятию верхнего уровня присваивается «удельный вес» a_i , $i \in I$. Полученный вектор a «весов» предприятий верхнего уровня сортируется по невозрастанию. Из компонент вектора b , соответствующих первым h компонентам вектора a , случайным образом выбирается q компонент,

которые приравниваются к единице (соответствующие предприятия открываются). Из остальных компонент b , отвечающих части (a_{h+1}, \dots, a_k) вектора a , случайным образом выбирается q компонент, которые зануляются (предприятия закрываются), и q компонент, которые приравниваются к единице (предприятия открываются). В результате получаем новое решение b' . Все векторы, найденные таким образом, составляют окрестность «взвешенных решений» вектора b . Заметим, что расстояние Хэмминга от любого решения этой окрестности до b не превышает $3q$.

При решении двухстадийной задачи размещения веса предприятий определялись следующим образом. Обозначим I_l — множество предприятий верхнего уровня, которые связаны с предприятием l нижнего уровня, т. е. $g_{il} = 1$, $i \in I_l$. Пусть множество L_i содержит предприятия нижнего уровня, связанные с предприятием i верхнего уровня, т. е. $g_{il} = 1$, $l \in L_i$. Определим число связей k_l предприятия l нижнего уровня с различными предприятиями верхнего уровня: $k_l = |I_l|$, $l \in L$. Тогда «вес» a_i предприятия i верхнего уровня равен $\sum_{l \in L_i} k_l$, $i \in I$.

В процессе построения алгоритма рассматривались различные последовательности и комбинации описанных окрестностей. Для отладки алгоритма использовалась библиотека тестовых задач [19], содержащая примеры различной сложности. В конечном итоге был найден лучший для данных тестовых примеров набор и порядок окрестностей. Для векторов y нижнего уровня использовались окрестности k -Flip, $k = 1, 2, 10$. Для векторов верхнего уровня рассматривались окрестности 1-Flip, 1-Swap и Лина — Кернигана.

С учётом того, что в двухстадийной задаче размещения множество предприятий делится на два подмножества, в предложенном алгоритме будет рассматриваться вектор открытия предприятий (z, y) , состоящий из $m + n$ компонент векторов z и y . С использованием окрестности для предприятий верхнего и нижнего уровней в этом решении изменяется первая (z', y) или вторая (z, y') часть. Для вектора $(\vec{0}, y')$ полагается $F(z, y) = \infty$, где $\vec{0} = (0, \dots, 0)$.

По векторам z можно восстановить вектор y , и наоборот. По компонентам вектора z компоненты вектора y строятся по правилу: если предприятие с номером l связано хотя бы с одним из открытых предприятий верхнего уровня, то полагаем $y_l = 1$, в противном случае $y_l = 0$. По компонентам вектора y формируется вектор z следующим образом: $z_i = 1$, $i \in I$, если $y_l = 1 \forall l \in L_i$.

Все новые решения, построенные указанными способами, допустимы. Кроме того заметим, что матрица назначений X по вектору z строится

аналитически. Для этого каждому клиенту j ставится в соответствие одно из открытых предприятий i верхнего уровня, которое обслуживает его с минимальными затратами; если таких предприятий несколько, то выбирается первое из них. Элемент матрицы x_{ij} полагается равным единице, а элементы $x_{i'j}$, $i' \neq i$, равными нулю.

Окрестностью $N_{k\text{-Flip}}(z, y)$ *нижнего уровня* булева вектора (z, y) будем называть множество векторов (z', y') , полученных из данного вектора заменой y на $y' \in N_{k\text{-Flip}}(y)$ и соответствующим изменением z .

Окрестностью $N_{k\text{-Swap}}(z, y)$ *нижнего уровня* булева вектора (z, y) будем называть множество векторов (z', y') , полученных из данного вектора заменой y на $y' \in N_{k\text{-Swap}}(y)$ и соответствующим изменением z .

Окрестностью $N^{k\text{-Flip}}(z, y)$ *верхнего уровня* булева вектора (z, y) будем называть множество векторов (z', y') , полученных из данного вектора заменой z на $z' \in N_{k\text{-Flip}}(z)$ и соответствующим изменением y .

Аналогично определяются остальные окрестности, используемые в предложенном алгоритме.

Значение $F(z, y)$, полученное в результате работы алгоритма, будем называть *рекордом алгоритма*, а вектор (z, y) — *рекордным решением*. Будем говорить, что алгоритм *выполнил одну итерацию*, если он просмотрел все окрестности из списка и не нашел решение, улучшающее рекордное значение целевой функции.

В данной статье в качестве начального решения алгоритма VNS всегда выбирался такой вектор (z, y) , что y и z нулевые, т. е. в начальный момент времени все предприятия нижнего и верхнего уровней закрыты. Критерием остановки служило предельное число итераций $\tau_{\max} = 200$. Обозначим через $\{N_r\}$, $r = 1, \dots, r_{\max}$, множество видов используемых окрестностей. Начальное решение $(z, y) = \vec{0}$, начальная итерация $\tau = 1$. Ниже дано пошаговое описание разработанного алгоритма.

СХЕМА АЛГОРИТМА VNS ДЛЯ ДВУХСТАДИЙНОЙ ЗАДАЧИ РАЗМЕЩЕНИЯ

ИТЕРАЦИЯ τ . ПРЕДВАРИТЕЛЬНЫЙ ШАГ. Полагаем $r := 1$, генерируем вектор z и строим по нему y .

ШАГ АЛГОРИТМА. Если $r = r_{\max} + 1$, переходим к следующей итерации $\tau := \tau + 1$, иначе выполняем следующие действия.

1. По некоторому правилу выбираем точку $(z', y') \in N_r(z, y)$.
2. Применяем некоторый метод локального поиска с начальным решением (z', y') и окрестностью «взвешенных решений». Обозначим через (z'', y'') найденный таким образом локальный оптимум.

3. Если $F(z'', y'') < F(z, y)$, полагаем $(z, y) := (z'', y'')$, $r := 1$ и идем на следующий шаг алгоритма; иначе полагаем $r := r + 1$.

Заметим, что в п. 1 правило выбора решения (z', y') зависит от вида окрестности.

4. Результаты вычислительного эксперимента

Для проведения вычислительного эксперимента использовалась библиотека тестовых примеров [19], содержащая классы задач различной сложности для алгоритмов ветвей и границ и локального поиска. Данная библиотека, в частности, содержит примеры двухстадийной задачи размещения, в которых затраты на открытие предприятий верхнего уровня равны нулю, т. е. $d_i = 0$, $i \in I$. Однако заметим, что при этом двухстадийная задача размещения по-прежнему остается NP -трудной и не превращается в простейшую задачу размещения. Отличие от ПЗР состоит в том, что потребителей обслуживают не предприятия нижнего уровня, а предприятия верхнего уровня. Как и в задаче (1)–(5), открытое предприятие верхнего уровня может участвовать в обслуживании клиентов только в том случае, если открыты все связанные с ним предприятия нижнего уровня. Исследования этой подзадачи и разработка вероятностных методов ее решения проводились ранее, например, в [3].

Используемая библиотека тестовых задач содержит пять классов: R4-Unif, R4-Eucl, 3-Galax, Gap-A, Gap-C. Для каждого класса создана серия из 30 примеров, в которых количество предприятий нижнего уровня $|L|$ равно 50, верхнего уровня $|I|$ равно 100, число потребителей $|J|$ равно 100. Для каждой задачи известно точное решение, полученное методом ветвей и границ [3]. Опишем классы тестовых задач.

СЕРИЯ 1, R4-Unif. Элементы матрицы транспортных затрат $T = (t_{ij})$ выбираются случайно с равномерным распределением из интервала $[0; 1000]$ независимо друг от друга. Каждое предприятие верхнего уровня связано с четырьмя предприятиями нижнего уровня, выбранными случайным образом, $\sum_{l \in L} g_{il} = 4$, $i \in I$. Стоимость открытия каждого предприятия нижнего уровня $c_l = 3000$, $l \in L$.

СЕРИЯ 2, R4-Eucl. Элементы матрицы транспортных затрат T удовлетворяют неравенству треугольника. Как и в первой серии, $\sum_{l \in L} g_{il} = 4$, $i \in I$; $c_l = 3000$, $l \in L$.

СЕРИЯ 3, 3-Galax. Множество из пятидесяти предприятий разбито на две части: 48 предприятий с низкой стоимостью 300 и 2 предприятия с высокой стоимостью 20000. Матрица $G = (g_{il})$ имеет специальную

структуру, $\sum_{l \in L} g_{il} = 5, i \in I$. Локальные оптимумы в данных примерах распадаются на три множества (галактики), расположенные достаточно далеко друг от друга.

СЕРИЯ 4, Гар-А. В каждом столбце матрицы T содержатся ровно 10 конечных элементов, значения которых получены с помощью псевдослучайной величины с равномерным распределением на множестве $\{0, 1, 2, 3, 4\}$. Начальные затраты на открытие каждого предприятия и «бесконечные» элементы в матрице T равны 3000; $\sum_{l \in L} g_{il} = 4, i \in I$.

СЕРИЯ 5, Гар-С. Построена аналогично четвертой серии. В отличие от класса Гар-А, в каждой строке и в каждом столбце матрицы T ровно 10 конечных элементов.

Вычислительный эксперимент состоял из двух этапов. На первом этапе проводилось тестирование разрабатываемых алгоритмов. На втором осуществлялось экспериментальное исследование лучшего варианта алгоритма локального поиска с чередующимися окрестностями. Все алгоритмы были реализованы на языке Visual C++ 6.0. Вычисления выполнялись на ЭВМ с процессором Athlon 3000+ с объемом оперативной памяти 512Мб.

Введем следующие обозначения: F_{opt} — оптимальное значение целевой функции; F_{min} — наилучшее значение целевой функции, полученное алгоритмом; ε — относительная погрешность решения (в процентах), $\varepsilon = \frac{F_{min} - F_{opt}}{F_{opt}} \cdot 100\%$; M_ε — точечная оценка математического ожидания относительной погрешности ε ; D_ε — точечная оценка дисперсии относительной погрешности ε ; t — среднее время выполнения одной итерации алгоритма в секундах.

На первом этапе проходил выбор вида окрестностей и порядок их чередования. Было получено две последовательности окрестностей $NS1$ и $NS2$. Последовательность $NS1$ включала пять окрестностей:

$$N_1 = N^{1\text{-Flip}}(z, y), N_2 = N_{1\text{-Flip}}(z, y), N_3 = N_{2\text{-Flip}}(z, y), \\ N_4 = N_{10\text{-Flip}}(z, y), N_5 = N^{1\text{-Swap}}(z, y).$$

В последовательности $NS2$ изменён порядок окрестностей и добавлена окрестность Лина — Кернигана:

$$N_1 = N^{1\text{-Flip}}(z, y), N_2 = N_{1\text{-Flip}}(z, y), N_3 = N_{2\text{-Flip}}(z, y), \\ N_4 = N^{1\text{-Swap}}(z, y), N_5 = N_{10\text{-Flip}}(z, y), N_6 = N^{LK}(z, y).$$

Окрестность Лина — Кернигана строилась с помощью окрестности 1-Swap. Добавление окрестности N^{LK} существенно увеличило (до шести раз) время счета, но улучшило качество решений только на серии Gap-A (см. табл. 1). Кроме того, анализ результатов данного эксперимента показал необходимость использования окрестности «взвешенных решений» в алгоритме простого локального поиска.

Т а б л и ц а 1. Точность решений с различными последовательностями окрестностей

Серия	3-Galax		Gap-A		Gap-C	
	$NS1$	$NS2$	$NS1$	$NS2$	$NS1$	$NS2$
opt	30	26	11	10	10	10
$\varepsilon < 1\%$	30	30	19	21	19	18

Для сокращения трудоемкости одного шага алгоритма VNS представляется целесообразным просматривать точки не из всей окрестности, а из некоторого её подмножества N' . Мощность этого подмножества также определялась на данном этапе. При использовании второго типа последовательности лучшие результаты были получены при $|N'| = 50$. Далее в эксперименте участвовал первый тип последовательности $NS1$ со значением $|N'| = 10$ в первой и четвертой окрестностях и $|N'| = 50$ в окрестности «взвешенных решений», остальные окрестности просматривались полностью.

На втором этапе проводилось экспериментальное исследование алгоритма локального поиска с чередующимися окрестностями с лучшими настройками параметров. Для сравнения результатов его работы использовался алгоритм имитации отжига (SA) [9, 12]. Отличительной чертой алгоритма имитации отжига является возможность шагов «назад», реализованная с помощью вероятностного перехода к новому решению. Стоит также отметить, что для SA доказана асимптотическая сходимость [15]. Алгоритмы SA хорошо показали себя при решении ряда задач дискретной оптимизации (см., например, [8–10, 18]). В эксперименте использовались варианты алгоритмов имитации отжига для двухстадийной задачи размещения, предложенные авторами и показавшие хорошие результаты с точки зрения точности получаемых решений и времени счета. Алгоритмы запускались по одному разу. Один запуск VNS состоял из 200 итераций, за один запуск SA 200 раз достигал температуры остывания.

Во время проведения вычислительного эксперимента каждым из предложенных алгоритмов было решено по 150 тестовых задач. Алгоритм

VNS нашел 111 оптимальных решений, SA — 112. Алгоритмы VNS и SA получили соответственно по 128 и 134 решений с относительной погрешностью менее одного процента.

Для исследуемых алгоритмов сложность задач с точки зрения нахождения оптимального решения увеличивалась с ростом номера серии. Классы R4-Unif и R4-Eucl оказались простыми для обоих алгоритмов. В каждой задаче было найдено оптимальное решение. Несмотря на то, что задачи серии 3-Galax являются трудными для алгоритмов локального поиска, алгоритм с чередующимися окрестностями нашел оптимальное решение в каждой задаче. Алгоритм имитации отжига не нашел оптимум на 9 примерах, но относительные погрешности на этих задачах не превысили 1%.

Четвертый и пятый классы задач оказались заметно сложнее. Наиболее трудным для алгоритма VNS оказался класс Gap-A. Для него наилучшие результаты показал алгоритм имитации отжига. Алгоритм локального поиска с чередующимися окрестностями нашел 20 решений с относительной погрешностью менее 1%, а алгоритм имитации отжига 27 решений.

Т а б л и ц а 2. Точность решений за 200 итераций

Серия	3-Galax		Gap-A		Gap-C	
	VNS	SA	VNS	SA	VNS	SA
opt	30	21	11	20	10	11
$\varepsilon < 1\%$	30	30	19	27	19	17

В табл. 2 приведено количество оптимальных и близких к ним решений ($\varepsilon \leq 1\%$) для последних трех классов задач. Данные об относительной погрешности найденных решений для всех классов задач приведены в табл. 3.

Т а б л и ц а 3. Значения погрешностей вычислений

Серия	VNS			SA		
	M_ε	D_ε	ε_{\max}	M_ε	D_ε	ε_{\max}
R4-Unif	0,000	0,000	0,000	0,000	0,000	0,000
R4-Eucl	0,000	0,000	0,000	0,000	0,000	0,000
3-Galax	0,000	0,000	0,000	0,057	0,018	0,487
Gap-A	1,853	7,202	8,658	0,427	1,675	4,310
Gap-C	1,785	6,828	7,953	1,988	7,981	11,920

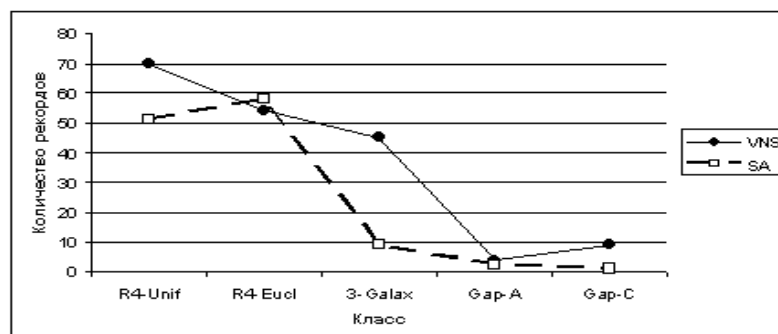


Рис. 1. Число рекордов

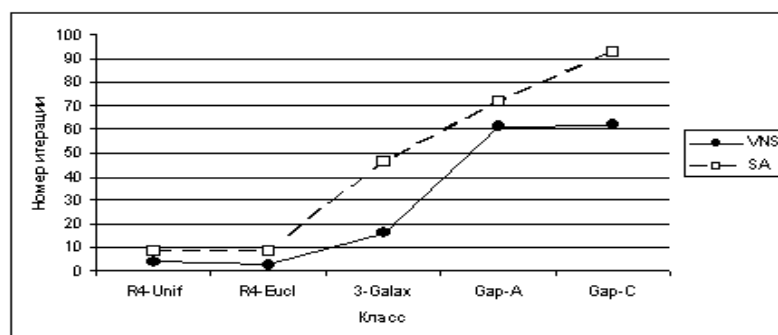


Рис. 2. Номер рекордной итерации

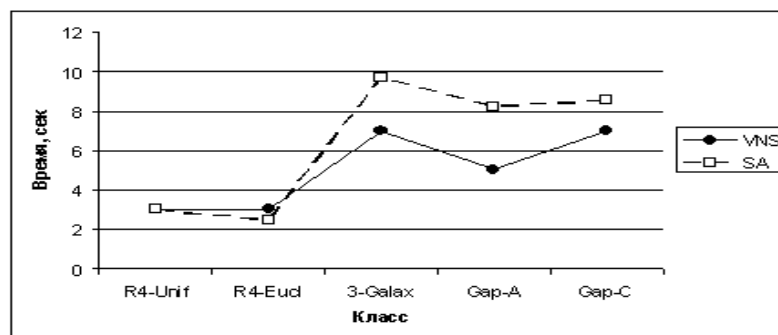


Рис. 3. Время нахождения рекорда

В целом следует отметить, что алгоритм с чередующимися окрестностями чаще находил рекордные решения и тратил на это меньшее число итераций. На рис. 1 изображено количество находений оптимальных решений для каждой серии. Так, на классе R4-Unif алгоритм VNS нашел оптимум в среднем на 71 итерации, алгоритм SA — на 58 итерации.

ях. На рис. 2 показано, на какой итерации впервые был найден рекорд алгоритма. На рис. 3 приведено время первого нахождения рекордного решения.

На графиках хорошо прослеживается тенденция ухудшения показателей алгоритмов с увеличением сложности задач.

Интересным также представляется вопрос о качестве работы алгоритмов с чередующимися окрестностями и имитации отжига при изменении числа итераций в одном запуске. В табл. 4 содержатся данные о числе найденных оптимальных решений при различных значениях этого параметра.

Т а б л и ц а 4. Число оптимальных решений

Серия	VNS						SA					
	5	10	50	100	150	200	5	10	50	100	150	200
R4-Unif	23	28	30	30	30	30	16	22	30	30	30	30
R4-Eucl	24	30	30	30	30	30	19	25	29	30	30	30
3-Galax	18	20	27	29	30	30	6	7	15	18	20	21
Gap-A	1	2	7	9	10	11	2	3	9	14	18	20
Gap-C	1	3	4	8	9	10	0	0	3	6	8	11

В описанном выше эксперименте алгоритмы затрачивали разное время на решение задач. Поэтому были проведены численные расчеты, в которых каждому алгоритму отводилось 10 минут на поиск решения. Полученные результаты отражены в табл. 5.

Т а б л и ц а 5. Точность решения за 10 мин

Серия	3-Galax		Gap-A		Gap-C	
	VNS	SA	VNS	SA	VNS	SA
opt	29	16	10	10	5	5
$\varepsilon < 1\%$	30	30	17	22	13	8

Таким образом, на основании экспериментальных исследований можно сделать вывод, что при различных условиях алгоритм локального поиска с чередующимися окрестностями показывает лучшие результаты на всех классах, кроме Gap-A, по сравнению с алгоритмом имитации отжига.

Заключение

В статье продолжены исследования, связанные с вопросами применения алгоритма локального поиска с чередующимися окрестностями для

задач дискретной оптимизации. Разработаны алгоритмы решения двухстадийной задачи размещения, экспериментально исследовано их поведение при различных наборах окрестностей и значениях параметров на трудных классах тестовых задач. Найдены последовательности окрестностей, при которых алгоритм показывает лучшие результаты на различных структурах данных.

В целом представленные в статье результаты экспериментальных исследований предложенных вариантов алгоритмов показали возможность их использования для сложных в вычислительном отношении задач и примеров большой размерности. В дальнейшем представляется интересным теоретическое исследование алгоритмов указанного вида, а также их применение при разработке точных алгоритмов решения дискретных задач оптимального размещения.

ЛИТЕРАТУРА

1. Береснев В. Л. Дискретные задачи размещения и полиномы от булевых переменных. — Новосибирск: Изд-во Ин-та математики, 2005. — 408 с.
2. Береснев В. Л., Гимади Э. Х., Дементьев В. Т. Экстремальные задачи стандартизации. — Новосибирск: Наука, 1978. — 335 с.
3. Гончаров Е. Н., Кочетов Ю. А. Поведение вероятностных жадных алгоритмов для многостадийной задачи размещения // Дискрет. анализ и исслед. операций. Сер. 2. — 1999. — Т. 6, № 1. — С. 12–32.
4. Колоколов А. А., Леванова Т. В., Лореш М. А. Алгоритмы муравьиной колонии для задач оптимального размещения предприятий // Омский науч. вестн. — 2006. — № 4 (38). — С. 62–67.
5. Кочетов Ю. А. Вероятностные методы локального поиска для задач дискретной оптимизации // Дискретная математика и ее приложения: Сборник лекций молодежных научных школ по дискретной математике и ее приложениям. — М.: МГУ, 2001. — С. 84–117.
6. Кочетов Ю. А., Младенович Н., Хансен П. Локальный поиск с чередующимися окрестностями // Дискрет. анализ и исслед. операций. Сер. 2. — 2003. — Т. 10, № 1. — С. 11–35.
7. Кузюрин Н. Н. Вероятностные приближенные алгоритмы в дискретной оптимизации // Дискрет. анализ и исслед. операций. Сер. 2. — 2002. — Т. 9, № 2. — С. 97–114.
8. Леванова Т. В., Лореш М. А. Алгоритмы муравьиной колонии и имитации отжига для задачи о p -медиане // Автоматика и телемеханика. — 2004. — № 3. — С. 80–88.
9. Dréo J., Pétrowski A., Siarry P., Taillard E. Métaheuristiques pour l'optimisation difficile. — Paris: Springer-Verl., 2006. — 369 p.
10. Johnson D. S., Aragon C. R., McGeoch L. A., Schevon C. Optimization by simulated annealing: an experimental evaluation; Part

- II, Graph coloring and number partitioning // European J. Oper. Res. — 1991. — Vol. 39. — P. 378–407.
11. **Hansen P., Mladenović N.** Variable neighborhood search // Handbook of Metaheuristics. — Chichester: Kluwer Acad. Publ., 2003. — P. 145–184.
12. **Kirkpatrick S., Gellat C. D., Vecchi M. P.** Optimization by simulated annealing // Science 220. — 1983. — P. 671–680.
13. **Krarup J., Pruzan P.** The simple plant location problem: survey and synthesis // European J. Oper. Res. — 1983. — Vol. 12, N 1. — P. 36–81.
14. **Local search in Combinatorial optimization** / Edited by E. Aarts and J. K. Lenstra. — New York: John Wiley & Sons, 1997.
15. **Lundy M., Meez A.** Convergence of an annealing algorithm // Math. Progr. — 1986. — V. 34. — P. 111–124.
16. **Mladenović N., Hansen P.** Variable neighborhood search // Computers Oper. Res. — 1997. — V. 24. — P. 1097–1100.
17. **Proceedings of the 18th Mini Euro Conference on Variable Neighborhood Search**, Puerto de la Cruz, Tenerife, Spain, 23.11.2005 - 25.11.2005. ISBN: 84-689-5679-1.
18. **Reeves C.** Modern heuristic techniques for combinatorial problems. — New York: John Wiley & Sons, 1993. — 320 p.
19. **Дискретные задачи размещения.** — www.math.nsc.ru/AP/benchmarks/

Леванова Татьяна Валентиновна,
e-mail: tlevanova@ofim.oscsbras.ru

Федоренко Анатолий Сергеевич,
e-mail: fas.omsk@mail.ru

Статья поступила
26 февраля 2008 г.

Переработанный вариант —
23 апреля 2008 г.