ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ
Сентябрь—октябрь 2012. Том 19, № 5. С. 63–82

УДК 519.8

# ЛОКАЛЬНЫЙ ПОИСК С ЧЕРЕДУЮЩИМИСЯ ОКРЕСТНОСТЯМИ ДЛЯ ЗАДАЧИ ДЖОНСОНА С ПАССИВНЫМ БУФЕРОМ *)

*П. А. Кононова, Ю. А. Кочетов*

**Аннотация.** Рассматривается задача теории расписаний потокового типа для двух машин с пассивной загрузкой буфера на второй машине. Для вычисления нижних оценок оптимума предложены четыре формулировки задачи в терминах целочисленного линейного программирования. Для нахождения верхних оценок разработаны три варианта метода локального поиска с чередующимися окрестностями. Наряду с известными полиномиальными окрестностями используется новая окрестность экспоненциальной мощности. Для проведения численных экспериментов построен новый класс тестовых примеров с известным значением оптимума. Результаты численных экспериментов на этом и других классах показали высокую эффективность разработанного подхода.

**Ключевые слова:** теория расписаний, локальный поиск, экспоненциальная окрестность.

## Введение

В [1, 10–12] рассмотрена задача, возникающая при создании электронно-цифровых библиотек и музеев. Медиа-объекты (файлы) загружаются из удалённой базы данных и затем воспроизводятся. Для каждого файла известны длительности загрузки и воспроизведения. При загрузке файл поступает в буфер транслирующего устройства и покидает его сразу после завершения воспроизведения. Размер буфера ограничен. Размеры файлов известны и пропорциональны временам загрузки. Воспроизведение файла не может начаться раньше окончания его загрузки. Если файл начал загрузку или воспроизведение, то этот процесс не прерывается. Требуется так задать порядки загрузки и воспроизведения файлов, чтобы минимизировать время окончания воспроизведения последнего файла.

*) Исследование выполнено при финансовой поддержке Российского фонда фундаментальных исследований (проект 11–07–00474).

Различают два способа загрузки буфера [10, 11]: активный и пассивный. При пассивном способе нельзя начинать загрузку файла, если свободного пространства в буфере недостаточно для помещения файла целиком. При активном способе загрузки считается, что в каждый момент времени общий размер загруженных и частично загруженных файлов не должен превышать размера буфера. Оптимальное решение задачи с активным способом загрузки, очевидно, является нижней оценкой для задачи с пассивным. Если в задаче с пассивным способом разрешить прерывания загрузки файлов, то получится задача, эквивалентная задаче с активным способом загрузки буфера [11]. Обе задачи NP-трудны в сильном смысле. Задача с активным способом загрузки подробно рассматривалась в [1, 11], в нашей статье рассматривается только задача с пассивным способом загрузки буфера.

Для вычисления нижних оценок оптимума и исследования окрестностей большой мощности предложено четыре формулировки задачи в терминах целочисленного линейного программирования (ЦЛП). В первых трёх формулировках предполагается, что порядок загрузки файлов может не совпадать с порядком их воспроизведения, в четвёртой эти порядки совпадают. Для задачи с активным буфером в множестве оптимальных решений всегда можно найти решение с одинаковыми порядками загрузки и воспроизведения файлов [11]. Для пассивного буфера этот вопрос пока остаётся открытым. В четвёртой формулировке ЦЛП рассматривается более ограниченная задача, которая также NP-трудна в сильном смысле. Наличие одной перестановки существенно сокращает время решения задачи точными методами, например, пакетом CPLEX, но даже при таком упрощении расчёты занимают слишком много времени уже при 20 файлах. В связи с этим особое внимание уделяется разработке приближённых методов и, в частности, стохастическим методам локального поиска с чередующимися окрестностями [1, 10]. В данной работе разработаны три варианта метода локального поиска, в которых наряду с известными окрестностями полиномиальной мощности используется новая окрестность экспоненциальной мощности, ранее применявшаяся для потоковых задач теории расписаний без буфера [4]. Для проведения численных экспериментов получен новый класс тестовых примеров с известным значением глобального оптимума. Численные эксперименты на классе примеров, случайно сгенерированных или порождаемых тестами для задачи упаковки в контейнеры [5], показали высокую эффективность разработанного метода: средняя погрешность не превышает 4% от нижней границы, а на примерах с известным оптимумом — не более 1%, если

число файлов не превосходит 100.

### 1. Постановка задачи и её ЦЛП формулировки

Как отмечалось во введении, воспроизведение файла не может начаться раньше окончания его загрузки. Более того, порядок загрузки файлов может отличаться от порядка их воспроизведения. Поэтому весь процесс загрузки и воспроизведения файлов удобно представлять в виде последовательной обработки файлов на машинах $A$ и $B$. Машина $A$ выполняет загрузку файлов в буфер, $B$ — их воспроизведение. Далее файлы будем называть работами, как это принято в теории расписаний.

Пусть $I$ — множество работ. Предполагаем, что оно конечно и его мощность равна $n$. Через $a_i$ обозначим время выполнения работы $i$ на машине $A$ (длительность загрузки), а через $b_i$ — время выполнение этой работы на машине $B$ (длительность воспроизведения). Размер буфера обозначим через $\Omega$, понимая под этим суммарное время, необходимое для заполнения всего буфера файлами. Предполагаем, что работа не может начаться на машине $A$, если в буфере не хватает места для её полного размещения. Задача состоит в том, чтобы найти порядок выполнения работ на машинах $A$ и $B$ такой, что общее время выполнения всех работ минимально. Если игнорировать буфер или считать его размер достаточно большим, то получаем известную задачу Джонсона на двух машинах [7].

Приведём несколько постановок задачи в терминах целочисленного линейного программирования. Они будут отличаться по числу переменных и ограничений и, как следствие, по качеству нижних и верхних оценок оптимума.

Предположим, что величины $a_i$ и $b_i$ — целые числа и $\mathbb{T}$ — верхняя оценка общего времени выполнения всех работ. Для любых момента времени $t = 0, 1, \ldots, \mathbb{T}$ и работы $i$ считаем, что $u_{it} = 1$ ($v_{it} = 1$), если работа $i$ начинается в момент $t$ на машине $A$ ($B$), и $u_{it} = 0$ ($v_{it} = 0$) в противном случае. Заметим, что каждая работа на каждой машине начинает выполняться только в один момент времени:

$$\sum_{t=0}^{\mathbb{T}} u_{it} = \sum_{t=0}^{\mathbb{T}} v_{it} = 1, \quad i \in I.$$

Кроме того, пока на машине выполняется одна работа, другая не может начать своё выполнение:

$$\sum_{i \in I} \sum_{\tau=\max\{0,t-a_i\}}^{t} u_{i\tau} \leqslant 1, \quad \sum_{i \in I} \sum_{\tau=\max\{0,t-b_i\}}^{t} v_{i\tau} \leqslant 1, \quad t = 0, \ldots, \mathbb{T}.$$

Каждая работа сначала выполняется на машине $A$, затем на машине $B$:

$$\sum_{t=0}^{\mathbb{T}} tu_{it} + a_i \leqslant \sum_{t=0}^{\mathbb{T}} tv_{it}, \quad i \in I.$$

Работа не может начинаться на машине $A$, если в буфере недостаточно свободного места:

$$a_i u_{it} \leqslant \Omega - \sum_{j \neq i} a_j \left( \sum_{\tau=0}^{t} u_{j\tau} - \sum_{\tau=a_j}^{t-b_j} v_{j\tau} \right), \quad t = 0, \ldots, \mathbb{T}, \ i \in I.$$

Требуется найти минимум целевой функции $F_1$, которая определяет время завершения всех работ:

$$F_1 \geqslant \sum_{t=0}^{\mathbb{T}} (t + b_i) v_{it}, \quad i \in I.$$

Сформулированная задача имеет $2n(\mathbb{T}+1)$ переменных и $2\mathbb{T} + 2n + n\mathbb{T}$ линейных ограничений. Обозначим её через UV1. Заметим, что число ограничений можно сократить, изменив ограничение на буфер:

$$\sum_{i \in I} \left( \sum_{\tau=0}^{t} a_i u_{i\tau} - \sum_{\tau=0}^{t-b_i} a_i v_{i\tau} \right) \leqslant \Omega, \quad t = 0, \ldots, \mathbb{T}.$$

Число переменных осталось прежним, а число ограничений сократилось до $3\mathbb{T}+2n$. Такая замена может иметь как положительные, так и отрицательные последствия. В частности, наличие бóльшего числа ограничений может приводить к лучшей линейной релаксации, но более компактная запись может сократить общее время счёта благодаря меньшей размерности. Обозначим новую запись задачи через UV2, а её целевую функцию — через $F_2$.

В [6] для формулировки задач в терминах ЦЛП использовался другой подход, основанный на понятии точек событий. Такая точка $k$ из интервала от 0 до $\mathbb{T}$ соответствует либо началу выполнения какой-то работы, либо её окончанию. Если в какой-то момент времени происходит одновременно несколько событий, то будем считать, что каждому событию соответствует своя точка и эти точки совпадают по времени. Так как рассматривается только две машины, общее число точек $K$ равно $4n$.

Введём четыре группы переменных. Положим $x^s_{ik} = 1$ ($x^f_{ik} = 1$), если точка $k$ соответствует началу (окончанию) выполнения работы $i$ на

машине $A$. Аналогично задаются переменные $y_{ik}^s$ и $y_{ik}^f$ для начала и окончания выполнения работы на машине $B$.

Пусть $T_k$ — момент времени, соответствующий точке $k$. Будем считать, что

$$T_1 = 0, \quad T_k \leqslant T_{k+1}, \quad 1 \leqslant k \leqslant K-1.$$

Каждая работа имеет ровно четыре точки события:

$$\sum_{k=1}^{K} x_{ik}^s = \sum_{k=1}^{K} x_{ik}^f = \sum_{k=1}^{K} y_{ik}^s = \sum_{k=1}^{K} y_{ik}^f = 1, \quad i \in I.$$

Каждая точка соответствует только одному событию:

$$\sum_{i \in I} \left( x_{ik}^s + x_{ik}^f + y_{ik}^s + y_{ik}^f \right) = 1, \quad k = 1, \ldots, K.$$

На каждой машине работа начинается, а потом заканчивается:

$$\sum_{l=1}^{k} x_{il}^s \geqslant \sum_{l=1}^{k} x_{il}^f, \quad \sum_{l=1}^{k} y_{il}^s \geqslant \sum_{l=1}^{k} y_{il}^f, \quad i \in I,\ k = 1, \ldots, K.$$

Каждая работа сначала выполняется на машине $A$, потом на машине $B$:

$$\sum_{l=1}^{k} x_{il}^f \geqslant \sum_{l=1}^{k} y_{il}^s, \quad i \in I,\ k = 1, \ldots, K.$$

Каждая машина в каждой точке выполняет не более одной работы:

$$\sum_{l=1}^{k} \sum_{i \in I} \left( x_{il}^s - x_{il}^f \right) \leqslant 1, \quad \sum_{l=1}^{k} \sum_{i \in I} \left( y_{il}^s - y_{il}^f \right) \leqslant 1, \quad k = 1, \ldots, K.$$

Если $k$ и $l$ — начало и окончание некоторой работы, то разность $T_l - T_k$ должна быть не меньше длительности этой работы:

$$T_l - T_k \geqslant a_i - a_i\left(2 - x_{il}^f - x_{ik}^s\right), \quad T_l - T_k \geqslant b_i - b_i\left(2 - y_{il}^f - y_{ik}^s\right), \quad i \in I,\ 1 \leqslant k < l \leqslant K.$$

Размер буфера ограничен:

$$\sum_{l=1}^{k} \sum_{i \in I} \left( a_i x_{il}^s - a_i y_{il}^f \right) \leqslant \Omega, \quad k = 1, \ldots, K.$$

Требуется найти минимум целевой функции, которая совпадает с $T_K$.

Сформулированная задача также относится к классу задач целочисленного линейного программирования. Она имеет $(16n^2+4n)$ переменных и $(24n+12n^2+32n^3)$ линейных ограничений. Экспериментальные исследования такой формы записи показывают, что оценка линейного программирования часто оказывается равной нулю, что хуже даже тривиальной нижней оценки $\max\big\{\sum\limits_{i\in I} a_i, \sum\limits_{i\in I} b_i\big\}$. Столь грубая нижняя оценка связана в первую очередь с ограничениями на длину интервала $T_l - T_k$, которые при дробных переменных $x^f_{il}, x^s_{il}$ и $y^f_{il}, y^s_{il}$ часто вырождаются в очевидные неравенства $T_l - T_k \geqslant 0$ для $l > k$.

Эти неравенства можно усилить, воспользовавшись условием, что в каждый момент времени на машине выполняется не более одной работы. Тогда получаем следующие условия, где $1 \leqslant k < l \leqslant K$:

$$T_l - T_k \geqslant \sum_{i\in I} \big(a_i - a_i\big(2 - x^f_{il} - x^s_{ik}\big)\big), \quad T_l - T_k \geqslant \sum_{i\in I} \big(b_i - b_i\big(2 - y^f_{il} - y^s_{ik}\big)\big),$$

или в эквивалентной форме

$$T_l - T_k \geqslant \sum_{i\in I} a_i\big(x^f_{il} - (1 - x^s_{ik})\big), \quad T_l - T_k \geqslant \sum_{i\in I} b_i\big(y^f_{il} - \big(1 - y^s_{ik}\big)\big).$$

Требование начать работу $i$ в точке $k$ содержится в выражении $\big(1 - x^s_{ik}\big)$ для машины $A$ и в выражении $\big(1 - y^s_{ik}\big)$ — для $B$. Это условие можно задать и другим способом, запретив в точке $k$ события на другой машине $(1 \leqslant k < l \leqslant K)$:

$$T_l - T_k \geqslant \sum_{i\in I} a_i\Big(x^f_{il} - \sum_{j\in I}\big(y^s_{jk} + y^f_{jk}\big)\Big), \quad T_l - T_k \geqslant \sum_{i\in I} b_i\Big(y^f_{il} - \sum_{j\in I}\big(x^s_{jk} + x^f_{jk}\big)\Big).$$

Дальнейшее усиление этих неравенств связано с желанием учесть несколько работ, попадающих в интервал $[T_k, T_l]$. Суммарная длительность работ, заканчивающихся в этом интервале, определяется суммой $\sum\limits_{q>k}^{l}\sum\limits_{i\in I} a_i x^f_{iq}$ для машины $A$ и аналогичной суммой для машины $B$. Однако первая из этих работ могла начаться до момента $T_k$. Удалив её из суммы, получаем новую нижнюю оценку на длину интервала $(1 \leqslant k < l \leqslant K)$:

$$T_l - T_k \geqslant \sum_{i\in I}\left(\sum_{q>k}^{l} a_i x^f_{iq} - a_{\max}\big(y^s_{ik} + y^f_{ik}\big)\right),$$

$$T_l - T_k \geqslant \sum_{i\in I}\left(\sum_{q>k}^{l} b_i y_{iq}^f - b_{\max}\bigl(x_{ik}^s + x_{ik}^f\bigr)\right),$$

где $a_{\max} = \max\limits_{i\in I} a_i$, $b_{\max} = \max\limits_{i\in I} b_i$. Полученную формулировку задачи обозначим через EP (event point), а соответствующую целевую функцию — через $F_{\mathrm{EP}}$.

Наконец, последняя формулировка связана с предположением об одинаковом порядке выполнения работ на обеих машинах. Такой вариант задачи часто называют *перестановочным*. Для задачи с активным буфером среди оптимальных решений всегда найдётся решение с указанным свойством [1]. Для пассивного буфера этот вопрос пока остаётся открытым. Тем не менее задача с пассивным буфером даже при таком упрощении остаётся NP–трудной в сильном смысле [11].

Пусть последовательность $\sigma(i)$, $i = 1, \ldots, n$, задаёт порядок выполнения работ. Введём булевы переменные $x_{ij}$, которые, как и $\sigma$, задают порядок выполнения работ. Положим $x_{ij} = 1$, если работа $i$ выполняется $j$-й по порядку, т. е. $i = \sigma(j)$, и $x_{ij} = 0$ в противном случае. Тогда

$$\sum_{l=1}^{n} x_{lj} = \sum_{k=1}^{n} x_{ik} = 1, \quad i, j = 1, \ldots, n.$$

Переменные $x_{ij}$ позволяют вычислить для машины $A$ длительность $j$-й по порядку работы в перестановке $\sigma$. Обозначим эту величину через $p^a_{\sigma(j)}$. Аналогичную величину для машины $B$ обозначим через $p^b_{\sigma(j)}$. Тогда

$$p^a_{\sigma(j)} = \sum_{i=1}^{n} a_i x_{ij}, \quad p^b_{\sigma(j)} = \sum_{i=1}^{n} b_i x_{ij}, \quad j = 1, \ldots, n.$$

Введём дополнительные переменные $s^a_{\sigma(j)}$ и $s^b_{\sigma(j)}$, которые будут определять начало выполнения $j$-й по порядку работы на машинах $A$ и $B$ соответственно. Значения этих переменных определяются из следующих очевидных соотношений:

$$s^a_{\sigma(1)} = 0,$$

$$s^a_{\sigma(j+1)} \geqslant s^a_{\sigma(j)} + p^a_{\sigma(j)}, \quad j = 1, \ldots, n-1,$$

$$s^b_{\sigma(j+1)} \geqslant s^b_{\sigma(j)} + p^b_{\sigma(j)}, \quad j = 1, \ldots, n-1,$$

$$s^b_{\sigma(j)} \geqslant s^a_{\sigma(j)} + p^a_{\sigma(j)}, \quad j = 1, \ldots, n.$$

Запишем условие ограниченности буфера. Пусть булева переменная $d^i_{jk}$ равна 1, если работа $i$ выполняется $k$-й по порядку и начинает загружаться в буфер до окончания выполнения $j$-й по порядку работы на машине $B$. В противном случае полагаем $d^i_{jk}$ равной 0. Используя эти переменные, можно получить ограничение на размер буфера:

$$\sum_{k=j}^{n}\sum_{i=1}^{n} a_i d^i_{jk} \leqslant \Omega, \quad j = 1, \ldots, n.$$

Для вычисления переменных $d^i_{jk}$ необходимо сравнивать величины $s^a_{\sigma(k)}$ и $s^b_{\sigma(j)} + p^b_{\sigma(j)}$. Для этих целей введём булеву переменную $c_{jk}$, равную 1, если $s^b_{\sigma(j)} + p^b_{\sigma(j)} > s^a_{\sigma(k)}$, и 0 в противном случае. Тогда переменные $c_{jk}$ определяются с помощью следующих неравенств:

$$\mathbb{T}c_{jk} \geqslant s^b_{\sigma(j)} + p^b_{\sigma(j)} - s^a_{\sigma(k)} + 1, \quad j = 1, \ldots, n,\ k = j, \ldots, n,$$

$$-\mathbb{T}(1 - c_{jk}) \leqslant s^b_{\sigma(j)} + p^b_{\sigma(j)} - s^a_{\sigma(k)}, \quad j = 1, \ldots, n,\ k = j, \ldots, n.$$

Кроме того, переменные $c_{jk}$ и $d^i_{jk}$ связаны соотношением

$$d^i_{jk} \geqslant c_{jk} + x_{ik} - 1, \quad i, j = 1, \ldots, n,\ k = j, \ldots, n.$$

Заметим, что неравенства

$$s^a_{\sigma(j+1)} \geqslant s^b_{\sigma(j)} + p^b_{\sigma(j)} + p^a_{\sigma(j)} - \Omega, \quad j = 1, \ldots, n-1,$$

для активного буфера [10] справедливы и для пассивного. Добавим их для усиления нижней оценки линейного программирования. Полученную формулировку обозначим через PFS (Permutation Flow Shop), а соответствующую целевую функцию — через $F_{\mathrm{PFS}}$. Задача состоит в нахождении допустимого расписания, доставляющего минимум целевой функции

$$F_{\mathrm{PFS}} = s^b_{\sigma(n)} + p^b_{\sigma(n)}.$$

Полученная формулировка, как и предыдущие, относится к классу целочисленного линейного программирования. Она включает $(4n + 2n^2 + n^3)$ переменных и $(9n + 2n^2 + n^3)$ ограничений. Численные эксперименты показывают, что она действительно проще для численных методов и позволяет быстро находить точное решение задачи, когда другие формулировки требуют слишком больших вычислительных затрат. Тем не менее даже эта формулировка для $n \geqslant 20$ на случайно сгенерированных данных тратит больше суток машинного времени на персональном

компьютере DualCore 2,8Ghz при решении пакетом CPLEX 11. В связи с этим в следующих разделах основное внимание уделяется стохастическим методам локального поиска, позволяющим быстро находить приближённые решения с малой относительной погрешностью.

## 2. Окрестности

Методы локального поиска показали высокую эффективность при решении широкого круга NP–трудных задач комбинаторной оптимизации. Локальный поиск с чередующимися окрестностями [2] принадлежит к этому классу методов. Его основная идея состоит в систематическом чередовании окрестностей и соответствующем изменении ландшафта в ходе локального поиска. При правильном выборе окрестностей этот метод гарантирует в асимптотике получение точного решения задачи [3].

Выбор системы окрестностей является принципиально важным для методов локального поиска. От него зависит как трудоёмкость каждой итерации, так и относительная погрешность решений, получаемых при заданном числе итераций. Для каждой задачи можно предложить несколько разнотипных окрестностей разной мощности и, как следствие, получить разные множества локальных оптимумов. Ниже будет предложено пять разнотипных окрестностей для перестановок, одна из которых используется для этой задачи впервые.

Пусть перестановка $\sigma$ задаёт некоторое допустимое решение задачи. Определим следующие окрестности для этого решения.

Окрестность $N_1(\sigma)$ содержит все перестановки, получающиеся из $\sigma$ либо перемещением одной работы на новое место, либо выбором двух работ и заменой их друг на друга. Мощность окрестности — $O(n^2)$.

Окрестность $N_l(\sigma)$, $l > 1$, содержит все перестановки, получающиеся из $\sigma$ не более $l$ последовательными переходами к соседним решениям по окрестности $N_1(\sigma)$. Мощность этой окрестности зависит от параметра $l$ и оценивается величиной $O(n^{2l})$.

Назовём *блоком* несколько подряд идущих работ в заданной перестановке.

Окрестность $N^b(\sigma)$ содержит все перестановки, получающиеся из $\sigma$ выбором двух непересекающихся блоков, возможно, разных размеров и заменой одного на другой. Мощность окрестности — $O(n^4)$.

При построении следующих двух окрестностей используется идея Кернигана — Лина, оказавшаяся продуктивной не только при разбиении графов, но и в задаче о $p$–медиане [8], построении расписаний школьных занятий [9] и др.

Окрестность $KL_l(\sigma)$, $l > 1$, содержит все перестановки, получающиеся из $\sigma$ по следующему правилу.

Шаг 1. Найдём наилучшую перестановку $\sigma'$ в окрестности $N_1(\sigma)$. Пусть это решение получается либо перемещением работы с позиции $j'$, либо взаимной перестановкой работ в позициях $j'$ и $j''$.

Шаг 2. Положим $\sigma := \sigma'$.

Будем повторять $l$ раз шаги 1 и 2, используя позицию $j'$ или пару позиций $(j', j'')$ не более одного раза. В результате получим $l$ перестановок. Каждую из них будем называть *соседней для $\sigma$ в окрестности $KL_l(\sigma)$*. Заметим, что $KL_l$ является частью окрестности $N_l$, но для нахождения всех её $l$ элементов требуется $O(ln^2)$ операций.

Окрестность $KL_l^b(\sigma)$, $l > 1$, строится аналогично $KL_l(\sigma)$, но вместо окрестности $N_1$ используется $N^b$. Трудоёмкость вычисления её $l$ элементов оценивается величиной $O(ln^4)$.

Наконец, рассмотрим окрестность экспоненциальной мощности, наилучший элемент которой может быть найден методами целочисленного линейного программирования. Пусть $\sigma(k, h)$ — часть перестановки $\sigma$ с позициями от $k$-й до $(k + h)$-й.

Окрестность $N(\sigma, k, h)$ содержит все перестановки, получающиеся из $\sigma$ всевозможными изменениями в перестановке $\sigma(k, h)$ [4]. Её мощность равна $h!$. Наилучшее решение при заданных $k$ и $h$ можно получить, например, решая задачу PFS с дополнительным ограничением

$$x_{ij} = 1, \quad i = \sigma(j),\ j = 1, \ldots, k-1, k+h+1, \ldots, n.$$

Рассмотренные окрестности требуют больших затрат машинного времени при их использовании в локальном поиске. В связи с этим ниже рассматриваются их рандомизированные аналоги, имеющие существенно меньшую мощность.

Под окрестностью $N_q(\sigma)$ будем понимать часть окрестности $N_1(\sigma)$. Каждый элемент окрестности $N_1(\sigma)$ включается в окрестность $N_q(\sigma)$ с вероятностью $q$ независимо от других элементов. Аналогично определяется $N_q^b(\sigma)$ как случайно выбранная часть окрестности $N^b(\sigma)$. Окрестности $KL_{ql}(\sigma)$ и $KL_{ql}^b(\sigma)$ определяются по тем же правилам, что $KL_l(\sigma)$ и $KL_l^b(\sigma)$ с соответствующей заменой окрестности $N_1$ на $N_q$ и $N^b$ на $N_q^b$. Выбирая параметр $q$, можно сократить трудоёмкость локального поиска по таким окрестностям в среднем в $1/q$ раз.

Для окрестности $N(\sigma, k, h)$ такой приём не срабатывает. Поэтому точное решение соответствующей подзадачи можно заменить приближённым

решением, получаемым, например, методом поиска с запретами (Tabu Search). Для этого будем использовать окрестности $N_q$ и $N_q^b$, не меняя перестановки вне интервала $[k, k+h]$. В списке запретов хранятся позиции перемещаемых работ и начала блоков. В качестве критерия остановки используется общее число итераций, т. е. переходов в соседнее решение по указанным окрестностям.

## 3. Локальный поиск

Приведём несколько вариантов локального поиска с чередующимися окрестностями (VNS) для решения рассматриваемой задачи. Зададим параметр $l_{\max}$, определяющий систему окрестностей $N_l$, $l = 1, \ldots, l_{\max}$, и параметр $l_0$, определяющий мощность окрестности $KL_{l_0}^b$, а также параметр рандомизации окрестностей $q$. Тогда общая схема метода может быть представлена следующим образом.

Алгоритм VNS1$(l_{\max}, l_0, q)$

Шаг 1. Выбрать начальное решение $\sigma$.

Шаг 2. Пока не выполнен критерий остановки, делать следующее.

Шаг 2(a). Положить $l := 1$.

Шаг 2(b). Повторять пока $l \leqslant l_{\max}$.

Шаг 2(b)i. Выбрать решение $\sigma' \in N_l(\sigma)$ случайным образом.

Шаг 2(b)ii. Применить алгоритм локального спуска по $N_q$.

Шаг 2(b)iii. Применить алгоритм локального спуска по окрестности $KL_{ql_0}^b$. Пусть $\sigma^*$ — полученный локальный минимум.

Шаг 2(b)iv. Если $F(\sigma^*) < F(\sigma)$, то $\sigma := \sigma^*$ и перейти на шаг 2(a), иначе положить $l := l + 1$.

Шаг 3. Предъявить решение $\sigma$ как результат работы алгоритма.

Начальное решение выбирается случайным образом. В качестве критерия остановки используется общее время счёта. В методе VNS1 не используется окрестность $N(\sigma, k, h)$. Естественно было бы включить процедуру локального улучшения по этой окрестности в случае, когда по другим окрестностям не удаётся улучшить рекордное решение. Такая модификация метода опробована, но оказалась неэффективной.

Наряду с данным методом логично рассмотреть аналогичный вариант, поменяв ролями окрестности $N_l(\sigma)$ и $KL_{ql_0}^b$. Тогда общая схема метода может быть представлена следующим образом.

Алгоритм VNS2($l_{\max}, l_0, q$)

Шаг 1. Выбрать начальное решение $\sigma$.

Шаг 2. Пока не выполнен критерий остановки, делать следующее.

Шаг 2(a). Положить $l := 1$.

Шаг 2(b). Повторять пока $l \leqslant l_{\max}$.

Шаг 2(b)i. Выбрать решение $\sigma' \in KL^b_{ql}$ случайным образом.

Шаг 2(b)ii. Применить алгоритм локального спуска по окрестности $N_q$.

Шаг 2(b)iii. Применить алгоритм локального спуска по окрестности $KL_{ql_0}$. Пусть $\sigma^*$ — полученный локальный минимум.

Шаг 2(b)iv. Если $F(\sigma^*) < F(\sigma)$, то $\sigma := \sigma^*$ и перейти на шаг 2(a), иначе положить $l := l + 1$.

Шаг 3. Предъявить решение $\sigma$ как результат работы алгоритма.

Применение рандомизированной окрестности $KL^b_{ql}$ на шаге 2(b)i позволяет избежать зацикливания даже при малых значениях параметра $l$. Несмотря на малую мощность этой окрестности каждый раз будут порождаться новые элементы, которые приведут к новым локальным оптимумам сначала по окрестности $N_q$, а затем по $KL_{ql_0}$. По тем же причинам, что в методе VNS1, здесь не используется окрестность $N(\sigma, k, h)$ экспоненциальной мощности.

В третьей схеме используется окрестность $N(\sigma, k, h)$. Основная проблема применения этой окрестности состоит в правильном выборе окна $[k, k + h]$, в котором следует проводить локальную оптимизацию. Для проверки перспективности заданного интервала применяется следующее правило: решается соответствующая задача линейного программирования, и полученная нижняя оценка $LP$ сравнивается с текущим значением целевой функции. Если эти величины совпали, то интервал является бесперспективным. В противном случае предпринимается попытка улучшить текущее решение за счёт перестановки работ в выбранном интервале. Начало интервала выбирается случайным образом. Так как некоторые интервалы могут оказаться (и часто оказываются) бесперспективными, последовательно выбирается не один, а $m$ интервалов. Для перспективных интервалов применяется поиск с запретами для нахождения наилучшей перестановки. Общая схема такого метода выглядит следующим образом.

Алгоритм VNS3($l_{\max}, m, h, q$)

Шаг 1. Выбрать начальное решение $\sigma$.

Шаг 2. Пока не выполнен критерий остановки, делать следующее.

Шаг 2(a). Положить $l := 1$.

Шаг 2(b). Повторять пока $l \leqslant l_{\max}$.

Шаг 2(b)i. Выбрать решение $\sigma' \in N_l(\sigma)$ случайным образом.

Шаг 2(b)ii. Применить локальный спуск по окрестности $N_q$. Обозначить найденный локальный минимум через $\sigma^*$.

Шаг 2(b)iii. Повторять $m$ раз следующие шаги.

Шаг 2(b)iiiA. Случайно выбрать начало интервала $k$.

Шаг 2(b)iiiB. Вычислить $LP = \mathrm{PFS}(\sigma^*, k, k+h)$.

Шаг 2(b)iiiC. Если $LP < F(\sigma)$, то $\sigma' = \mathrm{TabuSearch}(\sigma^*, k, k+h)$.

Шаг 2(b)iiiD. Если $F(\sigma') < F(\sigma^*)$, то $\sigma^* := \sigma'$.

Шаг 2(b)iv. Если $F(\sigma^*) < F(\sigma)$, то $\sigma := \sigma^*$ и перейти на шаг 2(a), иначе положить $l := l + 1$.

Шаг 3. Предъявить решение $\sigma$ как результат работы алгоритма.

Применение линейного программирования заметно сокращает число проверяемых интервалов. Попытки заменить линейное программирование и локальный поиск точным решением подзадачи на выбранном интервале не приводят к успеху. Подзадачи оказываются слишком сложными и трудоёмкими. Требуется либо специальная настройка точных методов с прерыванием вычислений, если подзадача не решается быстро, либо сокращение длины интервала.

## 4. Построение примеров с известным оптимумом

В [1] предложена идея генерации тестовых примеров с известным оптимумом, в основе которой лежит задача упаковки в контейнеры. Ниже предлагается другая идея. Произвольным образом выбираются длительности работ на обеих машинах, а затем находится минимальный размер буфера, при котором оптимальное решение задачи совпадает с нижней оценкой Джонсона.

Для реализации этой идеи достаточно сначала найти эту нижнюю оценку, удалив из рассмотрения буфер, а затем в множестве оптимальных решений такой упрощённой задачи найти оптимальное решение, требующее наименьший буфер. В [13] показано, что, переставляя соседние работы в оптимальной перестановке, можно получить любую другую оптимальную перестановку, при этом все промежуточные перестановки оптимальны. Таким образом, достаточно строить расписание работ, требующее наименьший буфер при оптимальной перестановке в задаче

Джонсона, и среди всех оптимальных перестановок находить наилучшую в этом смысле перестановку. Конструктивный алгоритм для достижения первой цели содержится в доказательстве следующего утверждения.

**Теорема 1.** *Пусть $\sigma$ — оптимальная перестановка работ для задачи Джонсона. Тогда для задачи с пассивным буфером можно за полиномиальное время найти минимальный размер буфера, при котором $\sigma$ останется оптимальной перестановкой.*

Доказательство. По перестановке $\sigma$ найдём критический путь в оптимальном расписании. Пусть работы $j_1, \ldots, j_k$ образуют начало этого пути на машине $A$, а работы $j_k, \ldots, j_n$ — конец пути на машине $B$. По работе $j_k$ происходит переход критического пути с первой машины на вторую. Так как путь критический, изменение расписания работ $j_1, \ldots, j_k$ на машине $A$ и работ $j_k, \ldots, j_n$ на машине $B$ ведёт к потере оптимальности. Следовательно, менять можно только расписание работ $j_1, \ldots, j_{k-1}$ на машине $B$ и работ $j_{k+1}, \ldots, j_n$ на машине $A$. Поставим эти работы на машине $B$ как можно раньше, а на машине $A$ как можно позже, не меняя длины расписания $Opt$.

Пусть, как и раньше, величины $s_i^a$, $s_i^b$ задают начала выполнения работ на машинах $A$ и $B$, а величины $c_i^a$, $c_i^b$ — окончание работ на этих машинах. Тогда

$$s_1^a = 0, \quad s_{i+1}^a = c_i^a, \quad s_{i+1}^b = \max\left\{c_i^b, c_{i+1}^a\right\}, \quad i = 1, \ldots, k-1,$$

$$c_n^b = Opt, \quad c_i^b = s_{i+1}^b, \quad c_i^a = \min\left\{s_{i+1}^a, s_i^b\right\}, \quad i = k, \ldots, n-1.$$

Для такого расписания вычислим требуемый размер буфера. Для каждого момента времени $t$ определим множество работ $I_t$, находящихся в этот момент в буфере: $I_t = \left\{i \mid s_i^a \leqslant t < c_i^b\right\}$. Тогда размер буфера $\Omega$ вычисляется как наибольший из потребовавшихся, т. е. $\Omega = \max\limits_{t>0} \sum\limits_{i \in I_t} a_i$. Покажем, что полученное значение является наименьшим из возможных.

Предположим, что это не так и можно как-то иначе перестроить расписание работ $j_1, \ldots, j_{k-1}$ на машине $B$ и $j_{k+1}, \ldots, j_n$ на машине $A$, что потребуется буфер меньшего размера. Тогда для момента времени $t_0$ такого, что $\Omega = \sum\limits_{i \in I_{t_0}} a_i$, множество $I_{t_0}$ должно содержать меньше работ. Другими словами, какие-то работы должны начаться позже $t_0$ на машине $A$, либо закончиться раньше $t_0$ на машине $B$, что по построению невозможно. Теорема 1 доказана.

**Замечание.** Доказанное утверждение можно обобщить на случай произвольной перестановки. Если $C_{\max}(\sigma)$ — длина расписания для $\sigma$

в задаче Джонсона, то за полиномиальное время можно построить расписание для задачи с пассивным буфером с той же длиной $C_{\max}(\sigma)$ и минимальным размером буфера. Доказательство проводится аналогично.

**Теорема 2.** *Задача минимизации буфера при заданной длине расписания* NP*-трудна в сильном смысле.*

Доказательство следует из NP-полноты следующей задачи распознавания [11]: *существует ли допустимое расписание заданной длины при фиксированном значении буфера?*

Если наименьший размер буфера найти трудно, то можно искать приближённое решение этой задачи, используя, например, метаэвристики. Полученное таким способом значение всё равно можно использовать при построении тестовых примеров. Нижняя оценка Джонсона по-прежнему будет достигаться.

Для получения приближённого решения снова воспользуемся схемой поиска с запретами. Для произвольной оптимальной перестановки работ в задаче Джонсона соседними решениями будем считать все оптимальные перестановки, получающиеся из данной выбором двух работ и заменой их друг на друга. В качестве списка запретов будем использовать позиции перемещаемых работ. Как и в случае метода с чередующимися окрестностями, будем применять рандомизацию окрестности. Критерий остановки — число выполненных итераций.

## 5. Численные эксперименты

Разработанные методы локального поиска запрограммированы на языке ПАСКАЛЬ (Delphi 7.0) и протестированы на трёх классах тестовых примеров:

случайно сгенерированных примерах, в которых величины $a_i$ и $b_i$, $i \in I$, выбираются независимо друг от друга с равномерным распределением из заданного интервала;

новом классе примеров, описанном в предыдущем разделе;

примерах для задачи упаковки в контейнеры.

Все расчёты проводились на персональном компьютере Pentium Dual Core 2,8Ghz RAM 2Gb с операционной системой Windows XP.

Для сравнения ЦЛП формулировок использовался первый класс примеров с $n = 10$, $a_i$, $b_i \in [1, 10]$, $i \in I$, $\Omega = 12$. На последних двух классах такой эксперимент не представляет интереса, так как для них разрыв целочисленности равен нулю. Табл. 1 содержит результаты расчётов для 15 примеров. Для каждой формулировки получены нижние оценки линейного программирования. Модель EP заметно выигрывает у моделей UV1

и UV2, но уступает модели PFS. Напомним, что в модели PFS используется только одна перестановка, так что сравнение моделей PFS и EP не совсем корректно. В табл. 2 приводятся результаты расчётов по исходным целочисленным моделям при ограничении в один час на каждый пример. Модель PFS позволяет решить все примеры, не требуя более двух секунд на каждый пример, в то время как другие модели не всегда дают точное решение задачи за отведённое время. Нули в колонках для значений целевой функции свидетельствуют о том, что не удалось получить даже допустимого решения. Интересно отметить, что ни в одном из примеров модель PFS не уступила по целевой функции своим конкурентам, хотя использование одной перестановки вместо двух могло бы привести к такому эффекту. Все попытки найти контрпример, в котором действительно требуется две перестановки, пока не увенчались успехом, что отчасти оправдывает стремление исследовать задачу в такой более простой форме. Далее оптимальное решение линейной релаксации для модели PFS будет использоваться в качестве нижней оценки при подсчёте погрешностей методов локального поиска.

Таблица 1

Различие между линейными релаксациями

| | UV1 | | UV2 | | EP | | PFS | |
|---|---|---|---|---|---|---|---|---|
| | $F_1$ | $t_1$ | $F_2$ | $t_2$ | $F_{\text{EP}}$ | $t_{\text{EP}}$ | $F_{\text{PFS}}$ | $t_{\text{PFS}}$ |
| 1 | 30,44 | 0:00:01 | 30,47 | 0:00:00 | 43 | 0:00:09 | 49 | 0:00:00 |
| 2 | 34,76 | 0:00:03 | 34,76 | 0:00:00 | 52,5 | 0:00:09 | 59 | 0:00:00 |
| 3 | 27,8 | 0:00:02 | 27,8 | 0:00:00 | 38,5 | 0:00:08 | 45 | 0:00:00 |
| 4 | 40,57 | 0:00:08 | 40,57 | 0:00:00 | 64,5 | 0:00:08 | 71 | 0:00:00 |
| 5 | 32,65 | 0:00:02 | 32,65 | 0:00:00 | 48,5 | 0:00:08 | 55 | 0:00:00 |
| 6 | 29,96 | 0:00:01 | 29,96 | 0:00:00 | 43 | 0:00:09 | 49 | 0:00:00 |
| 7 | 32,87 | 0:00:02 | 32,87 | 0:00:00 | 48,5 | 0:00:09 | 55 | 0:00:00 |
| 8 | 31,85 | 0:00:01 | 31,85 | 0:00:00 | 46,5 | 0:00:09 | 53 | 0:00:00 |
| 9 | 37,92 | 0:00:05 | 37,92 | 0:00:00 | 58,5 | 0:00:09 | 65 | 0:00:00 |
| 10 | 33,06 | 0:00:02 | 33,06 | 0:00:00 | 48,5 | 0:00:09 | 55 | 0:00:00 |
| 11 | 25,78 | 0:00:01 | 26,16 | 0:00:00 | 34,5 | 0:00:08 | 41 | 0:00:00 |
| 12 | 28,18 | 0:00:01 | 28,18 | 0:00:00 | 39 | 0:00:09 | 46 | 0:00:00 |
| 13 | 34,23 | 0:00:03 | 34,23 | 0:00:00 | 50,5 | 0:00:09 | 57 | 0:00:00 |
| 14 | 33,38 | 0:00:03 | 33,37 | 0:00:00 | 50,5 | 0:00:08 | 57 | 0:00:00 |
| 15 | 31,87 | 0:00:02 | 31,87 | 0:00:00 | 46,5 | 0:00:09 | 53 | 0:00:00 |
| среднее | 32,3547 | 0:00:02 | 32,3813 | 0:00:00 | 47,5333 | 0:00:09 | 54 | 0:00:00 |

Эффективность методов VNS1, VNS2 и VNS3 существенно зависит от параметра рандомизации $q$. Напомним, что этот параметр определяет долю просматриваемой окрестности. При малых значениях это-

го параметра сокращается время выполнения одной итерации, но возрастает вероятность пропустить оптимальное решение, даже находясь в непосредственной близости от него. На рис. 1 показано влияние параметра $q$ на результаты работы алгоритма VNS1. По оси абсцисс откладывается величина $\varepsilon$ — относительная погрешность получаемых решений, по оси ординат — число решений с погрешностью не более $\varepsilon$. Расчёты проводились при $q = 0{,}1; 0{,}2; \ldots, 1$. На рис. 1 представлены четыре графика для $q = 0{,}1; 0{,}2; 0{,}4; 1{,}0$. Величины $a_i$ и $b_i$ случайным образом с равномерным распределением выбираются из интервала от 300 до 700, $\Omega = 800$, $n = 100$, число испытаний алгоритма равно 50. Наилучшие результаты получены при $q = 0{,}1$. При увеличении числа итераций погрешность получаемых решений сокращается. Если же время счёта увеличить до 20 минут, то все получаемые решения совпадают с наилучшим найденным решением. Аналогичным образом ведут себя методы VNS2 и VNS3. Далее все расчёты проводятся при $q = 0{,}1$.

Т а б л и ц а 2

Различие между целочисленными моделями

| | UV1 | | UV2 | | EP | | PFS | |
|---|---|---|---|---|---|---|---|---|
| | $F_1$ | $t_1$ | $F_2$ | $t_2$ | $F_{\text{EP}}$ | $t_{\text{EP}}$ | $F_{\text{PFS}}$ | $t_{\text{PFS}}$ |
| 1 | 49 | 0:02:29 | 49 | 0:00:01 | 52 | 1:00:01 | 49 | 0:00:00 |
| 2 | *61* | 1:00:00 | 60 | 0:03:52 | 0 | 1:00:01 | 60 | 0:00:00 |
| 3 | 45 | 0:14:48 | 45 | 0:00:16 | 52 | 1:00:02 | 45 | 0:00:00 |
| 4 | 0 | 1:00:01 | 0 | 1:00:00 | 0 | 1:00:01 | 80 | 0:00:01 |
| 5 | 55 | 0:08:36 | 55 | 0:00:01 | 58 | 1:00:02 | 55 | 0:00:00 |
| 6 | 49 | 0:02:44 | 49 | 0:01:12 | 53 | 1:00:02 | 49 | 0:00:00 |
| 7 | 0 | 0:00:02 | 55 | 0:00:05 | 60 | 1:00:02 | 55 | 0:00:00 |
| 8 | 53 | 0:06:39 | 53 | 0:00:02 | 58 | 1:00:01 | 53 | 0:00:00 |
| 9 | 0 | 0:00:01 | 66 | 0:10:18 | 77 | 1:00:02 | 66 | 0:00:00 |
| 10 | 55 | 0:07:21 | 55 | 0:00:17 | 56 | 1:00:01 | 55 | 0:00:00 |
| 11 | *46* | 1:00:00 | 45 | 1:00:00 | 0 | 1:00:01 | 45 | 0:00:02 |
| 12 | 46 | 0:01:25 | 46 | 0:00:03 | 53 | 1:00:02 | 46 | 0:00:00 |
| 13 | 57 | 0:02:14 | 57 | 0:00:02 | 61 | 1:00:01 | 57 | 0:00:00 |
| 14 | 57 | 0:04:18 | 57 | 0:00:27 | 58 | 1:00:01 | 57 | 0:00:00 |
| 15 | 53 | 0:06:13 | 53 | 0:00:01 | 56 | 1:00:02 | 53 | 0:00:00 |
| среднее | 53,88 | 0:20:02 | 54,95 | 0:08:31 | 40,525 | 1:00:01 | 55,825 | 1,73611E-06 |

В табл. 3 приводятся результаты расчётов на данном классе тестовых примеров для трёх методов: VNS1, VNS2 и VNS3. Каждый из 40 примеров решался 50 раз каждым методом. В качестве критерия остановки использовалось время счёта, $l_{\max} = 15$, $l_0 = 30$. Для метода VNS3 полагалось $m = 5$, $h = 20$. Длина списка запретов равна 10, число итераций поиска с запретами — 200.

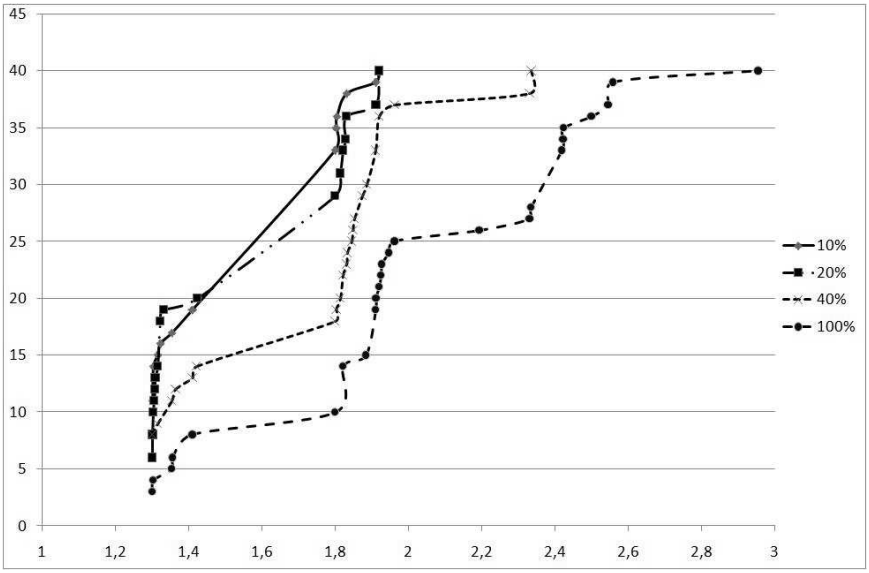


Рис. 1. Влияние параметра $q$

Таблица 3

Результаты для первого класса

| Метод | среднее | $\varepsilon$ | $\Delta$ | $\delta$ |
|---|---|---|---|---|
| VNS1 | 62551 | 2,98 | 183,05 | 0,021 |
| VNS2 | 62899 | 3,55 | 747,83 | 0,120 |
| VNS3 | 62766 | 3,34 | 325,5 | 0,037 |

Относительная погрешность методов рассчитывалась по формуле $\varepsilon = \frac{F_i - LB}{LB} 100\%$, где $F_i$ — значение целевой функции, полученное $i$-м методом, $LB$ — нижняя оценка. В табл. 3 приводятся средние значения целевой функции, средняя погрешность $\varepsilon$, её дисперсия $\delta$, а также величина $\Delta$ — наибольшая по всем примерам разница между максимальным и минимальным значением целевой функции. Полученные результаты свидетельствуют о том, что на данном классе примеров метод VNS1 имеет небольшое преимущество по средней погрешности и дисперсии. Метод VNS2 имеет наибольший разброс по целевой функции и наибольшую среднюю погрешность.

Второй класс тестовых примеров оказался несложным. Для $n = 50$ построено 10 примеров, каждый из которых решался 20 раз. Время поиска не превосходило 1 минуты. Табл. 4 содержит результаты расчётов. В скобках указано количество испытаний, в которых не найдено оптимального решения. Седьмой пример оказался наиболее трудным, но и на

нём только 5 раз из 20 не удалось найти оптимум методом VNS2, 2 раза из 20 — методом VNS1 и один раз из 20 — методом VNS3.

Таблица 4

Результаты для второго класса

| $n$ | VNS1 | $t_1$ | VNS2 | $t_2$ | VNS3 | $t_3$ |
|---|---|---|---|---|---|---|
| 1 | 1528 | 1 | 1528 | 1 | 1528 | 2 |
| 2 | 1487 | 1 | 1487 | 1 | 1487 | 1 |
| 3 | 1507 | 3 | 1507 | 1 | 1507 | 5 |
| 4 | 1532 | 1 | 1532 | 1 | 1532 | 2 |
| 5 | 1566 | 1 | 1566 | 1 | 1566 | 7 |
| 6 | 1523 | 1 | 1523 | 2 | 1523 | 5 |
| 7 | 1557(2) | 22 | 1557(5) | 34 | 1557(1) | 22 |
| 8 | 1515 | 1 | 1515 | 1 | 1515 | 1 |
| 9 | 1533 | 4 | 1533(3) | 16 | 1533 | 3 |
| 10 | 1549 | 1 | 1549 | 1 | 1549 | 1 |

В последнем эксперименте использовались трудные тестовые примеры для задачи упаковки в контейнеры, известные под названием триплеты [5]. В этих примерах оптимальное решение является разбиением $3k$ предметов на тройки. Каждая тройка соответствует одному контейнеру. Все контейнеры *плотно упакованы*, т. е. не содержат свободного места. Размер контейнера полагается равным 1000. Веса предметов $c_i$ выбираются из интервала $(250, 500)$. Положим $n = 4k$, $a_i = 1000$, $b_i = 0$ для $i \leqslant k$ и $a_i = 0$, $b_i = c_i$ для $k + 1 \leqslant i \leqslant 4k$. Тогда длина оптимального расписания при $\Omega = 1000$ равна $k\Omega$.

Для малой размерности $k = 10$, $n = 40$ метод VNS1 во всех примерах находит оптимальное решение. Метод VNS3 находит оптимальное решение только в 36 примерах из 50. При $k = 20$, $n = 80$ ни одному методу не удаётся найти оптимум за 30 минут поиска, но относительная погрешность всех получаемых решений не превосходит одного процента.

## ЛИТЕРАТУРА

1. **Кононова П. А.** Нижние и верхние оценки длины оптимального расписания презентаций медиа-объектов // Дискрет. анализ и исслед. операций. — 2012. — Т. 19, № 1. — С. 59–73.
2. **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—43.
3. **Brimberg J., Hansen P., Mladenović N.** Convergence of variable neighborhood search // Les Cahiers du GERAD G–2000–73. Montreal, Canada, 2000.

4. **Croce F. D., Grosso A., Salassa F.** A matheuristic approach for the total completion time two-machines permutation flow shop problem // Lect. Notes Comput. Sci. — 2011. — Vol. 6622. — P. 38–47.
5. **Falkenauer E.** A hybrid grouping genetic algorithm for bin packing // J. Heuristics. — 1996. — Vol. 2, N 1.— P. 5–30.
6. **Ierapetritou M. G., Floudas C. A.** Effective continuous-time formulation for shortterm scheduling: I. Multipurpose batch process // Ind. Eng. Chem. Res. — 1998. — Vol. 37. — P. 4341–4359.
7. **Johnson S. M.** Optimal two- and three-stage production scheduling with setup time included // Naval Res. Logist. — 1954. — Vol. 1. — P. 61–68.
8. **Kochetov Yu., Alekseeva E., Levanova T., Loresh M.** Large neighborhood local search for the $p$-median problem // Yugosl. J. Oper. Res. — 2005. — Vol. 15, N 1. — P. 53–63.
9. **Kochetov Yu., Kononova P., Paschenko M.** Formulation space search approach for the teacher/class timetabling problem // Yugosl. J. Oper. Res. — 2008. — Vol. 18, N 1. — P. 1–11.
10. **Kononov A., Kononova P., Hong J.-S.** Two-stage multimedia scheduling problem with an active prefetch model // Preprints of the 13th IFAC Symp. Information Control Problems in Manufacturing (Moscow, Russia, June 3–5, 2009). Moscow: Trapeznikov Institute of Control Sciences, 2009. — P. 1997–2002.
11. **Kononov A., Hong J.-S., Kononova P., Lin F.-C.** Quantity-based buffer-constrained two machine flowshop problem: active and passive prefetch models for multimedia applications // J. Sched. — 2012. — Vol. 15, N 4. — P. 487–497.
12. **Lin F.-C., Hong J.-S., Lin B. M. T.** A two-machine flow shop problem with processing time-dependent buffer constraints — an application in multimedia problem // Comput. Oper. Res. — 2009. — Vol. 36, N 4. — P. 1158–1175.
13. **Sevastianov S., Lin B. M. T.** Efficient enumeration of optimal and approximate solutions of the two-machine flow-shop problem // Preprint of 10th Workshop on Models and Algorithms for Planning and Scheduling Problems (Nymburk, Czech Republic, July 19–24, 2007). — Praha: Institute for Theoretical Computer Science (ITI) Charles University, 2011. — P. 177–179.

*Кононова Полина Александровна,*
e-mail: polinusik@gorodok.net
*Кочетов Юрий Андреевич,*
e-mail: jkochet@math.nsc.ru

Статья поступила
12 марта 2012 г.

Переработанный вариант —
14 августа 2012 г.