

УДК 519.8

НИЖНИЕ И ВЕРХНИЕ ОЦЕНКИ ДЛИНЫ ОПТИМАЛЬНОГО РАСПИСАНИЯ ПРЕЗЕНТАЦИЙ МЕДИА-ОБЪЕКТОВ *)

П. А. Кононова

Аннотация. Рассматривается задача теории расписаний потокового типа для двух машин с буфером на второй машине. Вводится понятие ограниченной задачи, и показана эквивалентность исходной и ограниченной задач. Исследуются нижние оценки оптимума. Установлено, что переход к ограниченной задаче приводит к улучшению нижних оценок. Для нахождения верхних оценок разработан итерационный метод локального поиска с чередующимися окрестностями. Используются четыре вида окрестностей, в том числе новая окрестность типа Кернигана — Лина. Численные эксперименты показали высокую эффективность предложенного подхода. Для сложных примеров метод локального поиска позволяет быстро найти точное решение или решение с малой относительной погрешностью.

Ключевые слова: теория расписаний, задача потокового типа, локальный поиск.

Введение

Рассмотрим следующую задачу, возникающую при создании электронно-цифровых библиотек и музеев [8]. Медиа-объекты (файлы) загружаются из удалённой базы данных и затем воспроизводятся. Для каждого файла заданы время 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 времени загрузки. Воспроизведение файла не может начаться раньше окончания его загрузки. Если файл начал загрузку или воспроизведение, то этот процесс не прерывается. Требуется так задать порядок загрузки и воспроизведения файлов, чтобы минимизировать время окончания воспроизведения последнего файла.

*) Исследование выполнено при финансовой поддержке Российского фонда фундаментальных исследований (проект 11-07-00474) и АВИП Рособразования (проект 2.1.1/3235).

Различают два способа загрузки. В [8] предполагается, что нельзя начинать загрузку файла, если свободного пространства в буфере недостаточно для помещения файла целиком. Такую задачу называют *задачей с пассивным буфером*. В [6] рассматривается другая модель, с более мягким ограничением на способ загрузки. Считается, что в каждый момент времени общий размер загруженных и частично загруженных файлов не должен превышать размера буфера. Другими словами, для очередного файла, если он не помещается в буфере целиком, можно рассчитать начало его загрузки, учитывая те файлы, которые вскоре будут удалены из буфера, чтобы буфер не был переполнен, а загрузка не прерывалась. Такую задачу называют *задачей с активным буфером*. Далее исследуется именно эта модель. В [7] показано, что задача с активным буфером эквивалентна задаче с пассивным буфером и разрешёнными прерываниями загрузки файлов. Известно [7], что эта задача является NP-трудной в сильном смысле даже для случая, когда время загрузки каждого файла не больше времени его воспроизведения.

Рассматриваемую задачу выбора порядка презентаций медиа-объектов можно представлять как задачу теории расписаний потокового типа для двух машин с буфером. Существует несколько различных понятий буфера для задач потокового типа. Обычно буфер представляет собой некоторое хранилище между машинами. Работа после выполнения на первой машине, если не может быть сразу выполнена на второй, попадает в буфер. Как только вторая машина освобождается, одна из работ покидает буфер и выполняется на второй машине. Буфер ограничивает количество работ, ожидающих освобождения второй машины. В общем случае эта задача NP-трудна, но для бесконечного и нулевого буфера известны полиномиальные алгоритмы [9].

В нашей статье рассматривается другой, более сложный случай. Во-первых, работа поступает в буфер по мере выполнения на первой машине, и размер занятого ей пространства пропорционален времени, потраченному первой машиной на эту работу к данному моменту. Работа покидает буфер только после окончания её выполнения на второй машине. Поэтому можно считать, что буфер находится на второй машине, а не между машинами, как в предыдущем случае. Другим отличием является то, что буфер ограничивает не число хранимых в нём работ, а их суммарный размер.

В [8] для задачи с пассивным буфером на второй машине предложен метод ветвей и границ. Он позволяет находить оптимальное решение для примеров малой размерности, не более 14 работ. В [7] для этого метода

предложены улучшенные нижние оценки, что позволяет увеличить размерность до 16 работ. Но эти нижние оценки не применимы для задач с активным буфером.

В [6] задача с активным буфером сформулирована в виде задачи целочисленного линейного программирования. Использование коммерческого программного обеспечения для решения задач целочисленного линейного программирования позволило находить оптимальное решение для примеров с 18 работами. Полученные результаты показывают, что нахождение точного решения является сложной вычислительной задачей даже для примеров небольшой размерности.

В данной работе для задачи с активным буфером рассматриваются примеры большей размерности и проводится сравнение нижних оценок оптимума с решениями, полученными методом локального поиска с чередующимися окрестностями. Вводится понятие ограниченной задачи. Для получения нижних оценок исследуется соответствующая задача линейного программирования и переход к задаче Джонсона путём удаления буфера. Эти нижние оценки несравнимы между собой, но переход к ограниченной задаче существенно сказывается на их повышении.

Метод локального поиска, применяемый для получения верхних оценок, использует четыре типа окрестностей, включая окрестность сдвига блоков в перестановке (под блоком понимают несколько элементов, идущих подряд в перестановке). С использованием такой окрестности получена новая окрестность типа Кернигана — Лина, которая позволяет заметно сократить погрешность получаемых решений по сравнению с решениями, найденными в [1, 6], и часто приводит к глобальному минимуму даже для сложных тестовых примеров. В ходе численных экспериментов наблюдался интересный эффект. Если длительности загрузки и воспроизведения файлов выбираются случайным образом с равномерным распределением из заданного интервала, то с ростом числа файлов задача не становится сложнее, а при большом числе файлов, пятьдесят и более, разрыв целочисленности отсутствует, нижняя оценка для ограниченной задачи совпадает с верхней и метод локального поиска даёт точное решение. Если взять трудные тестовые примеры для задачи упаковки предметов в контейнеры и адаптировать их под условия рассматриваемой задачи (размер буфера совпадает с размером контейнера), то разрыв целочисленности снова отсутствует, но найти оптимальное решение оказывается крайне сложно.

Работа организована следующим образом. В разд. 1 приводится математическая постановка задачи и обсуждаются возможности её решения

классическими методами целочисленного линейного программирования. Показано, что среди оптимальных решений задачи всегда найдётся решение, у которого порядки загрузки и воспроизведения файлов совпадают. Завершается раздел определением ограниченной задачи. В разд. 2 исследуются две нижние оценки (линейного программирования и Джонсона) и доказывается, что переход к ограниченной задаче не ухудшает этих оценок. В разд. 3 вводятся окрестности для локального поиска и приводится схема метода с чередующимися окрестностями. В разд. 4 обсуждаются результаты численных экспериментов. Показано влияние размера буфера на рост нижней оценки при переходе к ограниченной задаче. При больших размерах буфера задача вырождается, но при средних размерах относительно длины файлов рост нижней оценки может достигать 9%. Для случайно порождённых примеров показана разница между верхними и нижними оценками, отмечено их тотальное совпадение при большом числе работ. Для трудных тестовых примеров исследуется поведение метода локального поиска, частота получения оптимума и средняя погрешность. В разд. 5 даются заключение и основные выводы.

1. Математическая модель

Представим задачу в терминах теории расписаний. Рассмотрим две машины A и B . Каждому файлу сопоставим работу. Загрузке файла будет соответствовать выполнение работы на машине A . Длительность работы i на машине A обозначим через a_i и положим равной времени загрузки соответствующего файла. Воспроизведению файла соответствует выполнение работы на машине B . Длительность работы i на машине B обозначим через b_i и положим равной длительности воспроизведения файла. Так как воспроизведение файла не может начинаться до окончания загрузки файла, т. е. каждая работа сначала выполняется на машине A , а затем на машине B , полученная задача является задачей потокового типа для двух машин. Отличием данной задачи от известной задачи Джонсона является наличие буфера. Во время выполнения работы на машине A она загружается в буфер и освобождает его сразу после окончания выполнения на машине B . Обозначим через Ω размер буфера в секундах, т. е. время, необходимое для заполнения всего буфера работами. Тогда работа i занимает в буфере a_i секунд. Эту задачу будем называть *задачей Джонсона с буфером* и обозначать через ДБ.

Пусть последовательность $\sigma(i)$, $i = 1, \dots, n$, задаёт порядок выполнения работ. В общем случае порядок на машине A может отличаться от порядка на машине B . Ниже показано, что общий случай сводится

к частному случаю, когда порядки на обеих машинах совпадают. Введём переменные $x_{ij} \in \{0, 1\}$, которые так же, как и σ , задают порядок выполнения работ: $x_{ij} = 1$, если работа i выполняется j -й по порядку, т. е. $i = \sigma(j)$, и $x_{ij} = 0$ в противном случае. Тогда

$$\sum_{k=1}^n x_{kj} = \sum_{l=1}^n x_{il} = 1, \quad i, j = 1, \dots, n. \quad (1)$$

Переменные x_{ij} позволяют вычислить длительность работы на машине A для j -й по порядку работы в перестановке σ . Обозначим эту величину через $p_{\sigma(j)}^a$. Аналогичную величину для машины B обозначим через $p_{\sigma(j)}^b$. Тогда

$$p_{\sigma(j)}^a = \sum_{i=1}^n a_i x_{ij}, \quad j = 1, \dots, n, \quad (2)$$

$$p_{\sigma(j)}^b = \sum_{i=1}^n b_i x_{ij}, \quad j = 1, \dots, n. \quad (3)$$

Введём дополнительные переменные $s_{\sigma(j)}^a$ и $s_{\sigma(j)}^b$, которые будут определять начало выполнения j -й по порядку работы на машинах A и B соответственно:

$$s_{\sigma(1)}^a = 0, \quad (4)$$

$$s_{\sigma(j+1)}^a \geq s_{\sigma(j)}^a + p_{\sigma(j)}^a, \quad j = 1, \dots, n-1, \quad (5)$$

$$s_{\sigma(j+1)}^b \geq s_{\sigma(j)}^b + p_{\sigma(j)}^b, \quad j = 1, \dots, n-1, \quad (6)$$

$$s_{\sigma(j)}^b \geq s_{\sigma(j)}^a + p_{\sigma(j)}^a, \quad j = 1, \dots, n. \quad (7)$$

Условие ограниченности буфера влечёт [6]

$$s_{\sigma(j+1)}^a \geq s_{\sigma(j)}^b + p_{\sigma(j)}^b + p_{\sigma(j)}^a - \Omega, \quad j = 1, \dots, n-1. \quad (8)$$

Сформулированные условия задают множество допустимых расписаний. Задача состоит в нахождении допустимого расписания, доставляющего минимум целевой функции

$$F(\sigma) = s_{\sigma(n)}^b + p_{\sigma(n)}^b. \quad (9)$$

Задача (1)–(9) является задачей целочисленного линейного программирования. Такие задачи можно решать с помощью коммерческих пакетов.

Одним из лучших является пакет CPLEX [10]. К сожалению, численные эксперименты с этим пакетом уже для 20 работ требуют больших временных затрат. Для случайно сгенерированных исходных данных на этой размерности требуется больше суток машинного времени. В связи с этим актуальным является разработка приближённых методов, в частности, метаэвристик, а также нижних оценок оптимума.

1.1. Перестановочные расписания.

Теорема 1. *Среди оптимальных расписаний существует такое, в котором работы на машинах A и B выполняются в одинаковом порядке.*

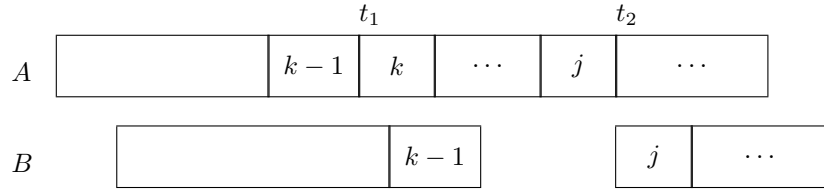


Рис. 1. Перестановочный приём

ДОКАЗАТЕЛЬСТВО. Пусть первые $k-1$ работ на машинах A и B в перестановке σ совпадают. Затем на машине A выполняется работа k , а на машине B — работа j . На машине A работа j выполняется после работы k . Обозначим через t_1 начало загрузки работы k на машине A , а через t_2 — время окончания загрузки работы j . Заметим, что с t_1 по t_2 на машине B выполняются работы, загруженные до t_1 , а все работы, загруженные после работы $k-1$, выполняются после t_2 . Поэтому изменение порядка работ на машине A в период (t_1, t_2) не влияет на порядок работ на машине B (рис. 1). Если в этот период нет простоев на машине A , то работы k и j на машине A можно поменять местами. Рассмотрим случай, когда есть простои на машине A . Пусть Δ — суммарная загрузка машины A в период (t_1, t_2) . Сдвинем все простои влево к t_1 , тогда вся загрузка будет проходить в последние Δ единиц времени. При этом длина расписания не изменится и буферное ограничение не будет нарушено. Поскольку загрузка происходит в каждый момент времени, работы k и j можно поменять местами. Теорема 1 доказана.

1.2. Ограниченная задача. Рассмотрим частный случай задачи ДБ, в котором для каждой работы i выполнено неравенство $a_i + b_i \leq \Omega$. Будем называть такую задачу *ограниченной задачей Джонсона с буфером* и обозначать через ОДБ.

Теорема 2. *Задачи ДБ и ОДБ эквивалентны.*

ДОКАЗАТЕЛЬСТВО ТЕОРЕМЫ 2 основано на следующем сведении задачи ДБ к ОДБ [6]. Пусть I — произвольный пример задачи ДБ. Построим пример \bar{I} задачи ОДБ с тем же числом работ и длительностями, равными $\bar{a}_i = a_i$ и $\bar{b}_i = \min(b_i, \Omega - a_i)$. Положим $\delta_i = b_i - \bar{b}_i$. Размер буфера оставим прежним. Тогда для целевой функции $\bar{F}(\sigma)$ задачи ОДБ выполнено равенство $F(\sigma) = \bar{F}(\sigma) + \sum_{i=1}^n \delta_i$ для любой перестановки σ .

2. Нижние оценки

Рассмотрим нижние оценки для задач ДБ и ОДБ.

2.1. Оценка линейного программирования. Заменим в (1)–(9) условие целочисленности переменных x_{ij} условием принадлежности их отрезку $[0, 1]$. Получим задачу линейного программирования. Оптимальное значение её целевой функции обозначим через LB_{lp} . Наряду с этой нижней оценкой рассмотрим оценку линейного программирования для задачи ОДБ. Обозначим её через \overline{LB}_{lp} .

Теорема 3. $LB_{lp} \leq \overline{LB}_{lp} + \sum_{i=1}^n \delta_i$.

ДОКАЗАТЕЛЬСТВО. Пусть \bar{x}_{ij} — оптимальное решение линейной релаксации задачи ОДБ. По нему однозначно определяются переменные \bar{p}_j^a , \bar{p}_j^b , \bar{s}_j^a и \bar{s}_j^b задачи ОДБ. Решение \bar{x}_{ij} также является допустимым для задачи ДБ, поскольку для \bar{x}_{ij} выполняется ограничение (1). Заметим также, что

$$p_j^a = \bar{p}_j^a, \quad p_j^b = \sum_{i=1}^n b_i \bar{x}_{ij} = \sum_{i=1}^n (\bar{b}_i + \delta_i) \bar{x}_{ij} = \bar{p}_j^b + \sum_{i=1}^n \delta_i \bar{x}_{ij}.$$

Введём величину $\chi_j = \sum_{i=1}^n \delta_i \bar{x}_{ij}$. Заметим, что

$$\sum_{j=1}^n \chi_j = \sum_{i,j=1}^n \delta_i \bar{x}_{ij} = \sum_{i=1}^n \delta_i.$$

По индукции докажем, что

$$s_j^a \leq \bar{s}_j^a + \sum_{k=1}^{j-1} \chi_k \quad \text{и} \quad s_j^b \leq \bar{s}_j^b + \sum_{k=1}^{j-1} \chi_k$$

для любого $j = 1, \dots, n$. Для $j = 1$ утверждение очевидно. Пусть утверждение справедливо для некоторого j . Покажем, что эти условия будут выполнены и для $j + 1$. Используя неравенства (1)–(9), получаем

$$\begin{aligned} s_{j+1}^a &= \max \{s_j^a + p_j^a, s_j^b + \bar{p}_j^b + p_j^a - \Omega + \chi_j\} \\ &\leq \max \{s_j^a + p_j^a, s_j^b + \bar{p}_j^b + p_j^a - \Omega\} + \chi_j \text{ (по предположению индукции)} \\ &\leq \max \left\{ \bar{s}_j^a + p_j^a + \sum_{k=1}^{j-1} \chi_k, \bar{s}_j^b + \bar{p}_j^b + p_j^a - \Omega + \sum_{k=1}^{j-1} \chi_k \right\} + \chi_j \\ &= \max \{ \bar{s}_j^a + p_j^a, \bar{s}_j^b + \bar{p}_j^b + p_j^a - \Omega \} + \sum_{k=1}^j \chi_k. \end{aligned}$$

Значит, $s_{j+1}^a \leq \bar{s}_{j+1}^a + \sum_{k=1}^j \chi_k$.

Аналогично докажем, что

$$\begin{aligned} s_{j+1}^b &= \max \{s_{j+1}^a + p_{j+1}^a, s_j^b + p_j^b\} \\ &\leq \max \left\{ \bar{s}_{j+1}^a + p_{j+1}^a + \sum_{k=1}^j \chi_k, \bar{s}_j^b + \sum_{k=1}^{j-1} \chi_k + \bar{p}_j^b + \chi_j \right\} \\ &= \max \{ \bar{s}_{j+1}^a + p_{j+1}^a, \bar{s}_j^b + \bar{p}_j^b \} + \sum_{k=1}^j \chi_k = \bar{s}_{j+1}^b + \sum_{k=1}^j \chi_k. \end{aligned}$$

В результате получим

$$\overline{LB}_{lp} + \sum_{i=1}^n \delta_i = \bar{s}_n^b + \bar{p}_n^b + \sum_{k=1}^n \chi_k \geq s_n^b + p_n^b \geq LB_{lp}.$$

Теорема 3 доказана.

2.2. Оценка Джонсона. Задача без буферного ограничения совпадает с классической задачей Джонсона. В литературе по теории расписаний для неё принято обозначение $F2||C_{\max}$. Пусть $s_j^a(I, \sigma)$ — время начала выполнения j -й по порядку работы на машине A в расписании σ для примера I . Напомним, что для задачи Джонсона существует оптимальное расписание, в котором порядок выполнения работ на машинах A и B совпадает и машина A работает без простоев. Тогда каждая перестановка σ задаёт единственное расписание и моменты начала выполнения работ задаются следующими формулами:

$$s_{j+1}^a(I, \sigma) = s_j^a(I, \sigma) + a_{\sigma(j)},$$

$$s_{j+1}^b(I, \sigma) = \max \{s_{j+1}^a(I, \sigma) + a_{\sigma(j+1)}, s_j^b(I, \sigma) + b_{\sigma(j)}\}.$$

Оптимальная перестановка в задаче $F2||C_{\max}$ может быть найдена за время $O(n \log n)$. Длина оптимального расписания будет нижней оценкой для исходной задачи с буфером. Назовём перестановку, соответствующую оптимальному решению задачи $F2||C_{\max}$, *перестановкой Джонсона*. Обозначим через LB_J нижнюю оценку, полученную алгоритмом Джонсона для задачи ДБ, а через \overline{LB}_J — нижнюю оценку для задачи ОДБ.

Теорема 4. $LB_J \leq \overline{LB}_J + \sum_i \delta_i$.

ДОКАЗАТЕЛЬСТВО. Пусть I — пример задачи ДБ, \bar{I} — соответствующий ему пример задачи ОДБ и σ — перестановка Джонсона для примера \bar{I} . Покажем по индукции, что

$$s_j^b(I, \sigma) \leq s_j^b(\bar{I}, \sigma) + \sum_{k=1}^{j-1} \delta_{\sigma(k)}.$$

Для $j = 1$ утверждение выполнено. Пусть утверждение справедливо для некоторого j . Тогда

$$\begin{aligned} s_{j+1}^b(I, \sigma) &= \max \{s_{j+1}^a(I, \sigma) + a_{\sigma(j+1)}, s_j^b(I, \sigma) + b_{\sigma(j)}\} \\ &\leq \max \left\{ s_{j+1}^a(I, \sigma) + a_{\sigma(j+1)}, s_j^b(\bar{I}, \sigma) + \sum_{k=1}^j \delta_{\sigma(k)} + \bar{b}_{\sigma(j)} \right\} \\ &\leq \max \{s_{j+1}^a(I, \sigma) + a_{\sigma(j+1)}, s_j^b(\bar{I}, \sigma) + \bar{b}_{\sigma(j)}\} + \sum_{k=1}^j \delta_{\sigma(k)} \\ &\leq s_{j+1}^b(\bar{I}, \sigma) + \sum_{k=1}^j \delta_{\sigma(k)}. \end{aligned}$$

Так как длина оптимального расписания примера I не превосходит длины расписания σ , получаем $LB_J \leq \overline{LB}_J + \sum_i \delta_i$. Теорема 4 доказана.

Оценки \overline{LB}_J и \overline{LB}_{lp} полиномиально вычислимы, однако для первой оценки имеется строго полиномиальный алгоритм Джонсона, а для второй требуется решение задачи линейного программирования. В этом смысле первая оценка предпочтительнее. Тем не менее нельзя утверждать, что одна из этих оценок лучше другой. В ходе численных экспериментов установлено, что они несравнимы между собой, но разница между ними относительно мала.

3. Поиск с чередующимися окрестностями

Пусть последовательность σ задаёт некоторое допустимое решение задачи. Определим следующие окрестности.

Окрестность $N_1(\sigma)$ содержит все последовательности, получающиеся из σ либо перемещением одной работы на новое место, либо взаимной перестановкой двух работ.

Окрестность $N_l(\sigma)$, $l > 1$, содержит все решения, получающиеся из данного не более l последовательными переходами к соседним решениям по окрестности $N_1(\sigma)$. Назовём *блоком* несколько подряд идущих работ последовательности σ .

Окрестность $N^b(\sigma)$ содержит все последовательности, получающиеся из σ взаимной перестановкой двух непересекающихся блоков, возможно разных размеров.

Окрестность $KL_k(\sigma)$, $k > 1$, является аналогом окрестности Кернигана — Лина [5] и строится следующим образом.

ШАГ 1. Найдём наилучшее решение σ' 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 шаги 1 и 2 не более k раз, используя позицию j' или пару (j', j'') не более одного раза. В результате получим набор последовательностей. Каждую из них будем считать соседней для исходного решения σ .

Вместо окрестности N_1 для построения окрестности Кернигана — Лина можно использовать окрестность N^b . Эта окрестность шире и позволяет находить решения, которые сложно получить используя окрестность N_1 . Окрестность N^b хорошо подходит для этой задачи, так как несколько работ могут создавать удачную подпоследовательность — блок, и расписание нужно изменять не для одной работы, а для всего блока. Просмотр окрестности N^b — трудоёмкая процедура, поэтому далее будем рассматривать блоки из не более чем трёх работ.

Поиск с чередующимися окрестностями (VNS) разработан Хансеном и Младеновичем в 1997 г. [2, 4]. Основная идея предлагаемого метода заключается в использовании разнородных окрестностей и их систематической смене в процессе локального поиска. Общая схема этого метода для решения данной задачи представлена ниже.

1. Найти начальное решение σ . Задать параметры l_{\max} , k_{\max} .

2. Пока не выполнен критерий остановки, делать следующее.
 - (а) Положить $l := 1$.
 - (б) Повторять пока $l \leq l_{\max}$.
 - i. Выбрать решение $\sigma' \in N_l(\sigma)$ случайным образом.
 - ii. Применить локальный спуск сначала по окрестности N_1 , а затем по окрестности $KL_{k_{\max}}$. Найденный локальный оптимум обозначить через σ^* .
 - iii. Если $F(\sigma^*) < F(\sigma)$, то $\sigma := \sigma^*$ и перейти на шаг 2(а), иначе положить $l := l + 1$.
 - (с) Применить локальный спуск по окрестности $KL_{k_{\max}}(\sigma)$, используя окрестность N^b вместо N_1 . Найденный локальный оптимум обозначить через σ^* . Если $F(\sigma^*) < F(\sigma)$, то $\sigma := \sigma^*$. Перейти на шаг 2(а).

Предъявить решение σ как результат работы алгоритма.

Начальное решение выбирается случайным образом. В качестве критерия остановки используется либо число полученных локальных оптимумов, либо общее время счёта.

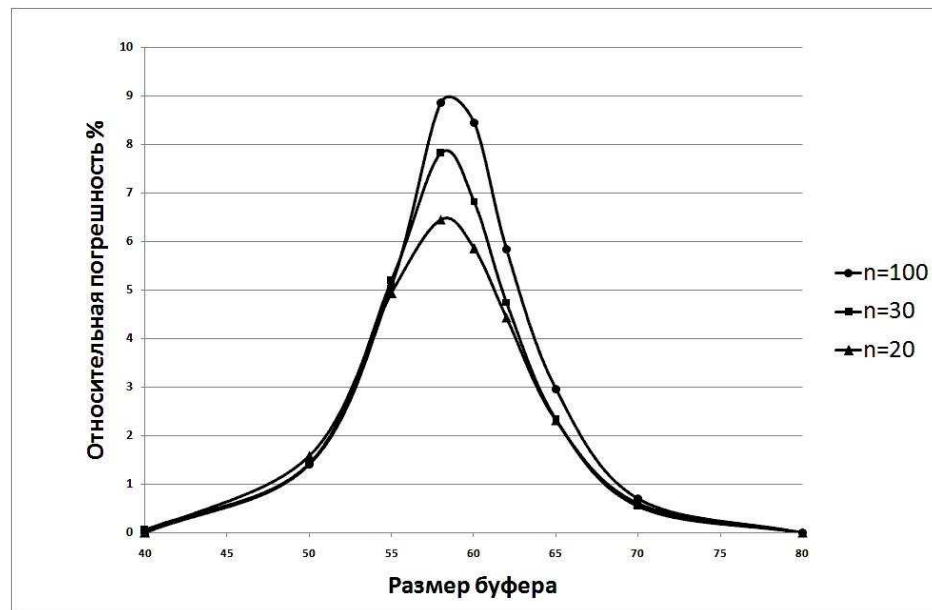


Рис. 2. Зависимость ε от размера буфера

Легко заметить, что с ростом размерности число несовпадений падает, а при $n = 50$ исчезает вовсе. При $n = 100$ и далее ситуация сохраняется. Получается удивительный эффект: с ростом размерности задачи она становится проще и при больших n легко решается точно. Алгоритм VNS быстро находит решение, значение которого совпадает с нижней оценкой. Такой эффект связан со способом генерации исходных данных. При большом разнообразии длительностей работ их легко компоновать под размер буфера. Изменив способ генерации исходных данных, можно получить трудные примеры, которые даже при отсутствии разрыва целочисленности будут представлять «головоломку».

В следующем эксперименте величины a_i и b_i строились на основе трудных тестов для задачи упаковки в контейнеры. Рассмотрим тесты, известные под названием триплеты [3]. Примеры в этих тестах характеризуются тем, что в каждом контейнере содержится ровно 3 предмета, и в оптимальном решении все контейнеры *плотно упакованы*, т.е. не содержат свободного места. Пусть $3k$ предметов имеют размеры $x_i \in (250, 500)$, а размер контейнера равен 1000. Положим $n = 4k$, $a_i = 1000$, $b_i = 0$ для $i \leq k$, $a_i = 0$ и $b_i = x_i$ для $k + 1 \leq i \leq 4k$, $\Omega = 1000$. Тогда оптимальное решение соответствует оптимальному решению задачи упаковки в контейнеры и имеет значение целевой функции $k\Omega$. Нижние оценки LB_{lp} и LB_J совпадают и равны оптимуму. Однако найти оптимум методами локального поиска трудно.

Для малой размерности, $k = 10$, $n = 40$, разработанный метод локального поиска в 44 примерах из 50 нашёл оптимальное решение, в остальных 6 примерах найденное решение отличалось от оптимального на 1. Алгоритм VNS без шага 2(с) за то же время получал результаты хуже [1]. Оптимальное решение найдено только в 10 примерах из 50, в остальных ответ отличался от оптимального в среднем на 4.

Для $k = 20$, $n = 80$ средняя погрешность составляет около 0,02%. Начальное решение выбиралось случайным образом. Время, отведённое для решения одной задачи на PC DualCore 2,8Ghz, равно одной минуте. Рис. 3 демонстрирует распределение значений целевой функции для решений, найденных алгоритмом VNS на одном из таких примеров за 100 независимых запусков. На оси ординат показаны значения целевой функции, на оси абсцисс — номер испытания алгоритма после сортировки по итоговым значениям целевой функции. Видно, что чаще всего алгоритм находит решения со значениями 20004 и 20005 что соответствует средней погрешности около 0,02%. Аналогичная картина наблюдается и на других примерах этого класса. Заметим, что CPLEX за 30 минут работы

на том же примере находит решение со значением 20891 и погрешностью около 4%. В [6] для тех же тестовых примеров среднее значение полученных решений составляет около 20006, при потраченном времени решения около полчаса на каждый из примеров.

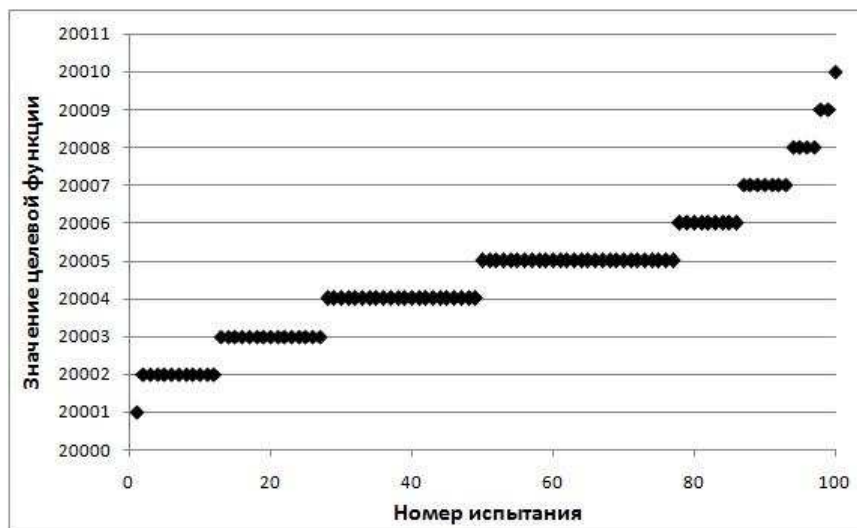


Рис. 3. Решения, найденные алгоритмом VNS

5. Заключение

Для задачи выбора порядка презентаций медиа-объектов получены алгоритмы вычисления нижних и верхних оценок оптимума. Показано, что переход к ограниченной задаче в случае активного буфера позволяет заметно улучшить нижние оценки. Для получения верхних оценок методами локального поиска предложена новая окрестность, позволяющая двигать блоки в перестановке. На её основе построена окрестность типа Кернигана — Лина, которая помогает заметно сократить погрешность получаемых приближённых решений задачи. В ходе численных экспериментов отмечено, что тестовые примеры со случайными длительностями работ из заданного интервала являются лёгкими, разрыв целочисленности часто отсутствует, а оптимальное решение находится достаточно быстро. Вместе с тем тестовые примеры, полученные на основе задачи упаковки в контейнеры, оказываются сложными несмотря на нулевой разрыв целочисленности.

Автор выражает искреннюю благодарность Ю. А. Кочетову за плодотворные дискуссии и полезные советы.

ЛИТЕРАТУРА

1. **Кононова П. А.** Алгоритм локального поиска для задачи выбора порядка презентаций медиа-объектов // Тр. ИВМиМГ. Информатика, 9. — Новосибирск: Изд-во ИВМиМГ СО РАН, 2009.— С. 177–182.
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. **Falkenauer E.** A hybrid grouping genetic algorithm for bin packing // J. Heuristics. — 1996. — Vol. 2, N 1. — P. 5–30.
4. **Hansen P., Mladenovich N.** Variable neighborhood search for the p -median problem // Locat. Sci. — 1997. — Vol. 5. — P. 207–226.
5. **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.
6. **Kononov A., Kononova P., Hong J.-S.** Two-stage multimedia scheduling problem with an active prefetch model // Preprints 13th IFAC Symp. Inf. Control Problems in Manufacturing (Moscow, Russia, June 3–5, 2009). — P. 1997–2002.
7. **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. Статья принята к печати. DOI: 10.1007/s10951-011-0235-z. <http://www.springerlink.com/content/82762q4066w16th8/>
8. **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.
9. **Papadimitriou C. H., Kanellakis P. C.** Flowshop scheduling with limited temporary storage // J. ACM. — 1980. — Vol. 27, N 3. — P. 533–549.
10. **CPLEX** <http://www-142.ibm.com/software/products/ru/ru/ilogcple/>

Кононова Полина Александровна,
e-mail: polinusik@gorodok.net

Статья поступила
5 апреля 2011 г.

Переработанный вариант —
14 июля 2011 г.