УЧЕНЫЕ ЗАПИСКИ КАЗАНСКОГО УНИВЕРСИТЕТА
*Том 154, кн. 3* Физико-математические науки *2012*

УДК 519.854.2

# О ДВУХ ВАРИАНТАХ МЕТОДА ВЕТВЕЙ И ГРАНИЦ ДЛЯ РЕШЕНИЯ ЗАДАЧИ МИНИМИЗАЦИИ СУММАРНОГО ВЗВЕШЕННОГО ЗАПАЗДЫВАНИЯ В КОНВЕЙЕРНЫХ СИСТЕМАХ

*И.К. Агапеевич, В.Р. Фазылов*

**Аннотация**

В статье предложены два варианта метода ветвей и границ для задачи минимизации суммарного взвешенного запаздывания в конвейерных системах, различающиеся тем, что в одном из них расписание строится в естественном порядке (сначала выбирается первая работа в расписании, затем вторая и т. д.), а в другом – в обратном порядке (сначала выбирается последняя работа в расписании, затем предпоследняя и т. д.). С помощью численного эксперимента показано, что эффективность методов существенно зависит от параметров задачи, легко вычисляемых по исходным данным, и предлагается критерий выбора для любой конкретной задачи более эффективного метода (из двух предложенных).

**Ключевые слова:** конвейерные системы, метод ветвей и границ, минимизация суммарного взвешенного запаздывания.

## Введение

Рассматривается простая конвейерная система из $m$ машин (см. [1, с. 136]). Требуется выполнить $n$ работ, каждая их которых представляет собой цепочку из $m$ операций. Все работы готовы к выполнению в момент времени 0, для каждой работы $i$ заданы директивный срок окончания выполнения $d_i$ и вес $w_i$, а для каждой операции $j$ работы $i$ – длительность выполнения $p_{ij}$. Допустимое расписание определяется следующими условиями:

1) каждая машина в любой момент времени может выполнять не более одной операции;

2) любая операция любой работы не может начаться ранее момента окончания предыдущей операции той же работы;

3) операции выполняются без прерываний и порядок прохождения работ по всем машинам одинаковый.

Пусть $C_i$ – момент завершения выполнения работы $i$ и $T_i = \max\{0, C_i - d_i\}$ – запаздывание работы $i$. Требуется построить расписание выполнения работ, обеспечивающее минимум суммарного взвешенного запаздывания $\sum\limits_{i=1}^{n} w_i T_i$.

Задача построения расписания, минимизирующего суммарное взвешенное запаздывание в конвейерной системе, представляет большой интерес для практики. Однако количество публикаций, посвященных этой теме, невелико (см., например, [2–7]). Известно, что задача минимизации суммарного взвешеннного запаздывания даже для одного исполнителя является NP-трудной [8]. Поэтому внимание исследователей преимущественно направлено на разработку различных эвристических методов, как, например, метод отжига [5], метод муравьиных колоний [9],

метод птичьих стай [6] и др. Тем не менее разработка точных методов решения рассматриваемой задачи представляет интерес, так как они дают возможность получить оптимальное решение задач небольших размерностей, оценить эффективность эвристических методов, а также классифицировать задачи по трудоемкости получения решения.

В статье предложены два варианта метода ветвей и границ, отличающихся порядком построения расписания: с начала (**Forward**) и с конца (**Backward**), и приведены результаты численного сравнения этих методов на задачах, полученных с помощью генератора задач из [10]. Результаты экспериментов показали, что трудоемкость получения решения любым вариантом метода ветвей и границ существенно зависит от параметров задачи, легко вычисляемых по исходным данным. Более того, на большинстве классов задач варианты метода ветвей и границ ведут себя разнонаправленно, и поэтому для конкретной задачи можно выбрать более эффективный вариант.

В теории расписаний большую роль играют регулярные критерии качества расписаний.

**Определение 1.** Критерий $f : R^n \to R$ будем называть регулярным, если для любых двух наборов $\{C_i\}_{i=1}^n$, $\{C'_i\}_{i=1}^n$, удовлетворяющих условию: $C_i \le C'_i$, $i = 1, 2, \ldots, n$, выполняется неравенство $f(C_1, \ldots, C_n) \le f(C'_1, \ldots, C'_n)$.

Приведенное понятие регулярного критерия является многомерным обобщением понятия неубывающей функции, и оно фактически эквивалентно определению регулярного критерия из [1, с. 13]. Но в [11, с. 24] регулярный критерий представлен как обобщение понятия возрастающей функции, причем несмотря на то что определение дано с ошибками, из текста определения следует, что авторы имели ввиду именно обобщение понятия возрастающей функции. Для устранения этой неточности введем понятие строго регулярного критерия, являющееся многомерным обобщением понятия возрастающей функции.

**Определение 2.** Критерий $f : R^n \to R$ будем называть строго регулярным, если для любых двух наборов $\{C_i\}_{i=1}^n, \{C'_i\}_{i=1}^n$, удовлетворяющих условиям: $C_i \le C'_i$ $i = 1, 2, \ldots, n$ и существует $k$: $C_k < C'_k$, выполняется неравенство $f(C_1, \ldots, C_n) < < f(C'_1, \ldots, C'_n)$.

Напомним, что простой машины называется искусственным, если для этой машины есть работа, ожидающая выполнения (см. [11, с. 38]).

**Теорема.** *Любая задача теории расписаний с регулярным критерием имеет оптимальное решение без искусственных простоев.*

Доказательство теоремы очевидно.

Так как критерий оптимальности расписания $\sum_{i=1}^{n} w_i T_i$ является регулярным (но не строго регулярным), то на основании теоремы и условия 3) допустимости расписания оптимальное расписание для рассматриваемой в работе задачи можно искать в классе перестановочных расписаний [11, с. 109].

## 1. Метод ветвей и границ

Как известно, метод ветвей и границ – это класс алгоритмов для решения различных задач дискретной оптимизации, имеющих, как правило, конечное множество допустимых решений. Общая схема метода включает четыре компонента (см. [13, с. 33]):

• конечное множество, охватывающее множество допустимых решений;

• иерархический способ разбиения охватывающего множества на подмножества вплоть до одноэлементных подмножеств, описываемый *деревом перебора решений*, и правило обхода этого дерева;

• способ получения нижней (для задачи минимизации) или верхней (для задачи максимизации) оценки целевой функции для подмножества решений, определенного любым узлом дерева перебора решений;

• быстрый способ получения хорошего допустимого решения (начального рекорда).

Суть метода ветвей и границ заключается в последовательном улучшении рекорда (наилучшего допустимого решения задачи на текущем этапе вычислений) путем обхода дерева перебора решений. Ключевым элементом метода является пренебрежение обходом поддеревьев, растущих из узлов дерева, в которых оценка целевой функции не лучше рекордного значения целевой функции. Заметим, что эффективность метода зависит от количества оптимальных или близких к оптимальным решений, способа разбиения множества, способа вычисления оценки целевой функции и качества начального рекорда. Заметим также, что в качестве начального рекорда можно выбрать любое допустимое решение, либо *фиктивный рекорд* – произвольное формальное решение со значением целевой функции, заведомо хуже оптимального.

**1.1. Дерево перебора решений и порядок его обхода.** Как было отмечено выше, класс перестановочных расписаний содержит оптимальное решение рассматриваемой задачи, поэтому его и выберем в качестве множества допустимых решений. В предлагаемых ниже вариантах метода ветвей и границ множество, охватывающее множество допустимых решений, совпадает с множеством перестановочных расписаний. Варианты метода ветвей и границ отличаются порядком построения расписания: вариант **Forward** строит расписание *с начала* (на первом уровне дерева определяется первая работа в расписании, на втором – вторая и т. д.), а вариант **Backward** – *с конца* (на первом уровне дерева определяется последняя работа в расписании, на втором – предпоследняя и т. д.).

Дерево перебора решений строится следующим образом. Множество всех перестановочных расписаний, соответствующее корню (узлу нулевого уровня) дерева, разбивается на $n$ подмножеств. Каждое из подмножеств характеризуется тем, что входящие в него расписания имеют одну и ту же работу на первом (**Forward**) или на последнем (**Backward**) месте. Таким образом, корень порождает $n$ узлов первого уровня. Далее каждое из множеств расписаний, соответствующих узлам первого уровня, разбивается аналогичным образом на $n-1$ подмножеств расписаний, соответствующих узлам второго уровня, и т. д.

Очевидно, что узел дерева перебора решений уровня $k$ соответствует кортежу $\pi^k$, представляющему собой список индексов $k$ первых или последних работ расписания. Корню дерева соответствует пустой кортеж $\pi^0$, а листу дерева – некоторый кортеж $\pi^{n-1}$, однозначно определяющий перестановочное расписание.

В обоих вариантах метода ветвей и границ узлы, полученные при очередном ветвлении, сортируются по неубыванию нижних оценок целевой функции для соответствующих подмножеств расписаний, а обход дерева осуществляется по правилу левостороннего обхода.

**1.2. Вычисление нижних оценок целевой функции при построении расписания с начала.** Пусть $\pi^k$ – список индексов $k$ первых работ расписания, $I$ – множество индексов работ, не включенных в список $\pi^k$. Ниже предлагается алгоритм вычисления нижней оценки целевой функции для заданного $\pi^k$.

**Метод Forward**

1. Построим расписание согласно списку $\pi^k$ и получим моменты завершения работ $C_i$, $i = \pi_1^k, \ldots, \pi_k^k$.

2. Вычислим нижние оценки моментов завершения работ $C_i'$, $i \in I$ при условии постановки каждой из непоставленных в расписание работ на $(k+1)$-е место.

3. Вычислим нижнюю оценку целевой функции:

$$F(\pi_k) = \sum_{i=1}^{k} w_{\pi_i^k} \max\{0, C_{\pi_i^k} - d_{\pi_i^k}\} + \sum_{i \in I} w_i \max\{0, C_i' - d_i\}.$$

**1.3. Вычисление нижних оценок целевой функции при построении расписания с конца.** Пусть теперь $\pi^k$ – список индексов $k$ последних работ расписания, $I$ – множество индексов работ, не включенных в список $\pi^k$. Ниже предлагается алгоритм вычисления нижней оценки целевой функции для заданного $\pi^k$.

**Метод Backward**

1. Вычислим нижние оценки моментов начала работы машин:

$$h_1 = 0, \quad h_j = h_{j-1} + \min_{i \in I} p_{ij-1}, \quad j = 2, \ldots, m.$$

2. Вычислим нижние оценки моментов освобождения машин после выполнения работ множества $I$ одним из трех способов:

1) простейший способ:

$$f_j = h_j + \sum_{i \in I} p_{ij}, \quad j = 1, \ldots, m;$$

2) так как момент освобождения машины $j$ при $j > 1$ не может быть меньше, чем $f_{j-1} + \min\limits_{i \in I} p_{ij}$, то предыдущие оценки можно усилить:

$$f_1 = h_1 + \sum_{i \in I} p_{i1}, \quad f_j = \max\{h_j + \sum_{i \in I} p_{ij}\,; f_{j-1} + \min_{i \in I} p_{ij}\}, \quad j = 2, \ldots, m;$$

3) обозначим через $S_{ijk}$ нижнюю оценку сдвига момента окончания работы $i$ на машине $k$ относительно момента окончания работы $i$ на машине $j$ $(j < k)$, вычисляемую по формуле $S_{ijk} = \sum\limits_{l=j+1}^{k} p_{il}$, а так как момент освобождения машины $j$ не может быть меньше, чем $f_l + \min\limits_{i \in I} S_{ilj}$ для любого $l$ $(l < j)$, то получим оценки:

$$f_1 = h_1 + \sum_{i \in I} p_{i1}, \quad f_j = \max\{h_j + \sum_{i \in I} p_{ij}; \max_{1 \le l \le j-1} \{f_l + \min_{i \in I} S_{ilj}\}\}, \quad j = 2, \ldots, m.$$

Заметим, что третий способ вычисления нижних оценок моментов освобождения машин после выполнения работ множества $I$ является естественным обобщением второго способа и идейно близок способу вычисления нижней оценки максимального момента окончания работ из [12]. Заметим также, что значения $\{\{\{S_{ijk}\}_{i=1}^{n}\}_{j=1}^{k-1}\}_{k=2}^{m}$ зависят лишь от исходных данных задачи, и их можно вычислить перед началом работы метода.

3. Начиная с моментов освобождения машин $\{f_j\}_{j=1}^{m}$, построим расписание согласно списку $\pi^k = \langle \pi_1^k, \ldots, \pi_k^k \rangle$, в результате чего получим нижние оценки моментов завершения работ $C_i$, $i = \pi_1^k, \ldots, \pi_k^k$. Вычислим нижнюю оценку целевой функции для множества расписаний, соответствующих списку $\pi^k$, одним из двух способов:

1) $F(\pi_k) = \sum\limits_{i=1}^{k} w_{\pi_i^k} \max\{0, C_{\pi_i^k} - d_{\pi_i^k}\}$;

2) $F(\pi_k) = \sum\limits_{i=1}^{k} w_{\pi_i^k} \max\{0, C_{\pi_i^k} - d_{\pi_i^k}\} + \min\limits_{i \in I}\{w_i \max\{0, f_m - d_i\}\}$.

Фактически описано шесть алгоритмов метода **Backward**, отличающихся способами вычисления нижних оценок моментов освобождения машин и нижних оценок целевой функции. Очевидно, что каждый следующий способ вычисления $\{f_j\}_{j=1}^m$ на шаге 2 метода дает результаты не хуже предыдущего, но является более трудоемким. Аналогичное замечание справедливо и относительно способов вычисления $F(\pi_k)$.

## 2. Численные эксперименты

**2.1. Генерация исходных данных задачи.** Для проведения численных экспериментов использовался генератор исходных данных задач [10], дающий задачи с заранее заданными значениями двух параметров – $TF$ (*tardiness factor*) и $RDD$ (*range of due dates*), – вычисляемых по формулам:

$$TF = 1 - \frac{\frac{1}{n}\sum\limits_{i=1}^{n} d_i - \frac{m-1}{nm}\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{m} p_{ij}}{\frac{1}{m}\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{m} p_{ij}}, \quad RDD = \frac{d_{\max} - d_{\min}}{\frac{1}{m}\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{m} p_{ij}},$$

где $d_{\max} = \max\limits_{1\le i\le n} d_i$ и $d_{\min} = \min\limits_{1\le i\le n} d_i$.

Поскольку далее параметры $TF$, $RDD$ будут использоваться и как параметры генератора, и как параметры задачи, для того чтобы различать их, будем обозначать параметры генератора как $TF_{\text{gen}}, RDD_{\text{gen}}$, а параметры сгенерированной задачи – $TF_{\text{pr}}, RDD_{\text{pr}}$.

Будем говорить, что сгенерированная задача принадлежит классу, определенному параметрами генератора $TF_{\text{gen}}, RDD_{\text{gen}}$, с точностью $\Delta$, если ее параметры $TF_{\text{pr}}, RDD_{\text{pr}}$ удовлетворяют условиям:

$$TF_{\text{pr}} \in [TF_{\text{gen}} - \Delta, TF_{\text{gen}} + \Delta], \quad RDD_{\text{pr}} \in [RDD_{\text{gen}} - \Delta, RDD_{\text{gen}} + \Delta].$$

### Алгоритм генерации исходных данных задачи

1. Выберем значения $n, m$, параметры $TF_{\text{gen}}, RDD_{\text{gen}}$ из интервала [0, 1] и параметр $\Delta > 0$.

2. Из целых чисел интервала [1, 10] случайно по равномерному закону выберем значения $\{w_i\}_{i=1}^n$.

3. Из целых чисел интервала [1, 100] случайно по равномерному закону выберем значения $\{\{p_{ij}\}_{j=1}^m\}_{i=1}^n$, вычислим $P = \sum\limits_{i=1}^{n}\sum\limits_{j=1}^{m} p_{ij}$.

4. Из целых чисел интервала

$$\left[\frac{P(m-1)}{nm} + \frac{P}{m}\left(1 - TF - \frac{RDD}{2}\right); \frac{P(m-1)}{nm} + \frac{P}{m}\left(1 - TF + \frac{RDD}{2}\right)\right]$$

случайно по равномерному закону выберем значения $\{d_i\}_{i=1}^n$.

5. Положим $d_i = \max\{d_i, \sum\limits_{j=1}^{m} p_{ij}\}$, $i = 1, \ldots, n$.

6. Вычислим параметры задачи $TF_{\text{pr}}$, $RDD_{\text{pr}}$. Если окажется, что $TF_{\text{pr}} \notin [TF_{\text{gen}} - \Delta, TF_{\text{gen}} + \Delta]$ или $RDD_{\text{pr}} \notin [RDD_{\text{gen}} - \Delta, RDD_{\text{gen}} + \Delta]$, то перейдем к п. 2, иначе останов: исходные данные задачи сгенерированы.

Табл. 1

$n = 12,\ m = 4,\ N = 100$

| $TF_{\text{gen}}$ | $RDD_{\text{gen}}$ | 1-1 | 2-1 | 3-1 |
|---|---|---|---|---|
| 0.1 | 0.1 | 1.00 / 101373 | 1.02 / 95233 | 1.09 / 90658 |
| 0.1 | 0.3 | 0.40 / 42830 | 0.42 / 41328 | 0.47 / 41169 |
| 0.1 | 0.5 | 0.24 / 26391 | 0.25 / 25432 | 0.27 / 25035 |
| 0.1 | 0.7 | 0.02 / 2096 | 0.02 / 1931 | 0.02 / 1834 |
| 0.1 | 0.9 | 0.02 / 2607 | 0.02 / 2389 | 0.02 / 2117 |
| 0.3 | 0.1 | 0.24 / 23285 | 0.25 / 22225 | 0.27 / 20768 |
| 0.3 | 0.3 | 0.13 / 12533 | 0.13 / 11670 | 0.15 / 11281 |
| 0.3 | 0.5 | 0.07 / 7260 | 0.07 / 6519 | 0.08 / 6072 |
| 0.3 | 0.7 | 0.06 / 5710 | 0.06 / 5313 | 0.07 / 4986 |
| 0.3 | 0.9 | 0.09 / 7886 | 0.09 / 7565 | 0.10 / 7027 |
| 0.5 | 0.1 | 0.17 / 14277 | 0.18 / 13423 | 0.20 / 12592 |
| 0.5 | 0.3 | 0.17 / 14459 | 0.18 / 13129 | 0.20 / 12351 |
| 0.5 | 0.5 | 0.15 / 12067 | 0.16 / 11346 | 0.18 / 10885 |
| 0.5 | 0.7 | 0.28 / 22875 | 0.30 / 21835 | 0.34 / 20841 |
| 0.5 | 0.9 | 0.34 / 27110 | 0.37 / 26023 | 0.42 / 25297 |
| 0.7 | 0.1 | 1.14 / 86267 | 1.23 / 81081 | 1.38 / 76292 |
| 0.7 | 0.3 | 1.16 / 87211 | 1.26 / 82886 | 1.45 / 79868 |
| 0.7 | 0.5 | 1.54 / 116679 | 1.67 / 111409 | 1.92 / 107235 |
| 0.7 | 0.7 | 1.45 / 107690 | 1.58 / 103133 | 1.82 / 100499 |
| 0.9 | 0.1 | 6.33 / 479005 | 6.94 / 464290 | 7.97 / 450690 |
| 0.9 | 0.3 | 4.11 / 300035 | 4.51 / 291930 | 5.22 / 284208 |

**2.2. Описание экспериментов.** Эксперименты проводились по следующей схеме:

– $TF_{\text{gen}}$, $RDD_{\text{gen}}$ табулировались от 0.1 до 0.9 с шагом $h$, равным 0.2, но на основании численных экспериментов, описанных в [10], генерация задач производилась лишь для пар $TF_{\text{gen}}$, $RDD_{\text{gen}}$, удовлетворяющих условию:

$$RDD_{\text{gen}} < \min\{1; 2.3 - 2 \cdot TF_{\text{gen}}\};$$

– для каждой пары $TF_{\text{gen}}$, $RDD_{\text{gen}}$ было сгенерировано $N$ задач с размерностями $n \times m$ ($n$ – количество работ, $m$ – количество машин).

Результаты экспериментов представлены в табл. 1–6. Значения в таблицах приведены в формате $A\,/\,B$, где $A$ – среднее время решения задачи (в секундах), $B$ – среднее количество просмотренных узлов.

Вычисления проводились на PowerBook G4 (1.67 GHz, 512 MB).

### Эксперимент 1

Целью первого эксперимента являлось сравнение предложенных в статье способов вычисления нижних оценок в методе **Backward**. Каждая задача была решена всеми шестью алгоритмами, отличающимися способами вычисления нижних оценок моментов освобождения машин и целевой функции. Алгоритмы метода **Backward** обозначены как "$X$-$Y$", где $X$ – номер способа вычисления нижней оценки моментов освобождения машин (см. п. 2 метода **Backward**), $Y$ – номер способа вычисления нижней оценки целевой функции (см. п. 3 метода **Backward**).

Из табл. 1 (в ней представлены алгоритмы, в которых нижняя оценка целевой функции вычислялась способом 1) видно, что алгоритм 1-1 для всех классов задач дает наилучший результат в смысле среднего времени решения задачи, тогда как алгоритм 3-1 – наилучший результат в смысле среднего количества просмотренных узлов.

Табл. 2

$n = 12,\ m = 4,\ N = 100$

| $TF_{\text{gen}}$ | $RDD_{\text{gen}}$ | 1-2 | 2-2 | 3-2 |
|---|---|---|---|---|
| 0.1 | 0.1 | 1.03 / 101125 | 1.05 / 94989 | 1.12 / 90301 |
| 0.1 | 0.3 | 0.42 / 42828 | 0.43 / 41312 | 0.48 / 41160 |
| 0.1 | 0.5 | 0.25 / 26316 | 0.26 / 25359 | 0.28 / 24965 |
| 0.1 | 0.7 | 0.02 / 2096 | 0.02 / 1931 | 0.02 / 1834 |
| 0.1 | 0.9 | 0.02 / 2607 | 0.02 / 2389 | 0.02 / 2117 |
| 0.3 | 0.1 | 0.23 / 21294 | 0.23 / 19881 | 0.27 / 19877 |
| 0.3 | 0.3 | 0.12 / 11460 | 0.13 / 11168 | 0.14 / 10907 |
| 0.3 | 0.5 | 0.08 / 7961 | 0.08 / 6937 | 0.09 / 6631 |
| 0.3 | 0.7 | 0.06 / 5717 | 0.06 / 5317 | 0.07 / 4981 |
| 0.3 | 0.9 | 0.09 / 7828 | 0.10 / 7383 | 0.11 / 6842 |
| 0.5 | 0.1 | 0.18 / 14229 | 0.18 / 13111 | 0.20 / 12176 |
| 0.5 | 0.3 | 0.18 / 13875 | 0.18 / 12450 | 0.20 / 11792 |
| 0.5 | 0.5 | 0.16 / 11951 | 0.17 / 11188 | 0.19 / 10740 |
| 0.5 | 0.7 | 0.29 / 22611 | 0.31 / 21303 | 0.34 / 20128 |
| 0.5 | 0.9 | 0.35 / 26462 | 0.38 / 25021 | 0.43 / 24132 |
| 0.7 | 0.1 | 1.11 / 80006 | 1.11 / 70862 | 1.17 / 63598 |
| 0.7 | 0.3 | 1.17 / 82004 | 1.20 / 75212 | 1.33 / 70676 |
| 0.7 | 0.5 | 1.57 / 110530 | 1.63 / 101716 | 1.80 / 95468 |
| 0.7 | 0.7 | 1.50 / 103482 | 1.55 / 93958 | 1.74 / 89508 |
| 0.9 | 0.1 | 5.43 / 379828 | 5.30 / 327479 | 5.74 / 299432 |
| 0.9 | 0.3 | 3.70 / 252273 | 3.67 / 221173 | 4.03 / 205961 |

Из табл. 2 (в ней представлены алгоритмы, в которых нижняя оценка целевой функции вычислялась способом 2) видно, что алгоритм 1-2 для большинства классов задач дает наилучший результат в смысле среднего времени решения задачи, за исключением классов с $TF = 0.9$, где лучшим оказался алгоритм 2-2. Наилучший результат в смысле среднего количества просмотренных узлов показал алгоритм 3-2.

В целом по результатам эксперимента видно, что наиболее простой алгоритм 1-1 для большинства классов задач дает лучшие результаты в смысле среднего времени решения задачи, за исключением следующих случаев:

1) для классов $(0.3; 0.1), (0.7; 0.1)$ лучшими оказались алгоритмы 1-2 и 2-2;

2) для класса $(0.3; 0.3)$ лучшим оказался алгоритм 1-2;

3) для классов $(0.9; 0.1), (0.9; 0.3)$ лучшим оказался алгоритм 2-2.

Видно также, что алгоритм 3-2 просматривает на 10–20% узлов меньше, чем другие алгоритмы.

### Эксперимент 2

Второй эксперимент был посвящен выявлению эффективности применения сортировки узлов по неубыванию нижних оценок целевой функции (см. п. 1.1). Обоими методами было решено по 100 задач различных размерностей для каждой пары параметров $TF$, $RDD$ как без сортировки, так и с предварительной сортировкой узлов. В результате проведенного эксперимента было выявлено, что сортировка узлов сокращает среднее время решения задач не менее чем вдвое.

### Эксперимент 3

Третий эксперимент проводился с целью сравнения методов построения расписания с конца и с начала. Каждая задача была решена двумя методами: **Backward** (лучшим для соответствующего класса алгоритмом) и **Forward**.

Табл. 3

$n = 12,\ m = 4,\ N = 100$

| $TF_{\text{gen}}$ | $RDD_{\text{gen}}$ | Backward | Forward |
|---|---|---|---|
| 0.1 | 0.1 | 1.00 / 101373 | 35.59 / 5727497 |
| 0.1 | 0.3 | 0.40 / 42829 | 18.78 / 2939693 |
| 0.1 | 0.5 | 0.24 / 26391 | 5.21 / 789569 |
| 0.1 | 0.7 | 0.02 / 2095 | 0.75 / 102734 |
| 0.1 | 0.9 | 0.02 / 2606 | 0.11 / 14694 |
| 0.3 | 0.1 | 0.23 / 19881 | 7.46 / 920219 |
| 0.3 | 0.3 | 0.12 / 11460 | 5.12 / 619750 |
| 0.3 | 0.5 | 0.07 / 7259 | 2.62 / 321121 |
| 0.3 | 0.7 | 0.06 / 5709 | 0.82 / 98397 |
| 0.3 | 0.9 | 0.09 / 7886 | 0.25 / 29494 |
| 0.5 | 0.1 | 0.17 / 14277 | 1.95 / 212231 |
| 0.5 | 0.3 | 0.17 / 14459 | 1.47 / 151503 |
| 0.5 | 0.5 | 0.15 / 12066 | 0.94 / 93741 |
| 0.5 | 0.7 | 0.28 / 22874 | 0.61 / 62777 |
| 0.5 | 0.9 | 0.34 / 27110 | 0.25 / 25541 |
| 0.7 | 0.1 | 1.11 / 70862 | 0.66 / 64960 |
| 0.7 | 0.3 | 1.16 / 87210 | 0.58 / 56578 |
| 0.7 | 0.5 | 1.54 / 116679 | 0.58 / 55622 |
| 0.7 | 0.7 | 1.45 / 107690 | 0.42 / 39338 |
| 0.9 | 0.1 | 5.30 / 327479 | 0.40 / 37572 |
| 0.9 | 0.3 | 3.67 / 221173 | 0.35 / 33790 |

Табл. 4

$n = 13,\ m = 4,\ N = 40$

| $TF_{\text{gen}}$ | $RDD_{\text{gen}}$ | Backward | Forward |
|---|---|---|---|
| 0.5 | 0.5 | 0.66 / 47521 | 8.77 / 884497 |
| 0.5 | 0.7 | 1.78 / 132084 | 2.31 / 201575 |
| 0.5 | 0.9 | 1.93 / 129900 | 1.53 / 131280 |
| 0.7 | 0.1 | 3.24 / 193158 | 4.52 / 426094 |
| 0.7 | 0.3 | 5.46 / 354248 | 3.23 / 278696 |
| 0.7 | 0.5 | 9.11 / 696781 | 1.92 / 158464 |

Табл. 5

$n = 13,\ m = 9,\ N = 40$

| $TF_{\text{gen}}$ | $RDD_{\text{gen}}$ | Backward | Forward |
|---|---|---|---|
| 0.1 | 0.7 | 2.27 / 87680 | 7.37 / 424348 |
| 0.1 | 0.9 | 4.72 / 198159 | 1.04 / 50518 |
| 0.3 | 0.1 | 6.78 / 253904 | 16.16 / 786554 |
| 0.3 | 0.3 | 19.77 / 772338 | 15.40 / 778159 |
| 0.3 | 0.5 | 6.91 / 261586 | 8.38 / 399909 |
| 0.3 | 0.7 | 14.22 / 545578 | 5.56 / 271307 |
| 0.3 | 0.9 | 8.84 / 324139 | 1.76 / 86190 |
| 0.5 | 0.1 | 17.81 / 669627 | 5.81 / 264634 |
| 0.5 | 0.3 | 19.18 / 693398 | 4.40 / 194186 |

В табл. 3 видна четкая тенденция зависимости трудоемкости решения задач для каждого метода от класса, которая подтверждается результатами экспериментов и на задачах других размерностей. Так, трудоемкость метода **Forward** падает с ростом $TF$, а для одинаковых значений $TF$ падает с ростом $RDD$. Для метода

Табл. 6

| $m/n$ | $(TF; RDD)$ | $m/n$ | $(TF; RDD)$ |
|---|---|---|---|
| $< 0.20$ | (0.9; 0.1) | 0.42–0.46 | (0.5; 0.5) |
| 0.20–0.26 | (0.7; 0.7) | 0.47–0.52 | (0.5; 0.3) |
| 0.27–0.32 | (0.7; 0.3) | 0.53–0.60 | (0.3; 0.9) |
| 0.33–0.37 | (0.5; 0.9) | 0.61–0.75 | (0.3; 0.7) |
| 0.38–0.41 | (0.5; 0.7) | $> 0.75$ | (0.1; 0.7) |

**Backward** свойственна другая тенденция и наиболее трудоемкими являются задачи классов, для которых $TF \in \{0.7, 0.9\}$. Более того, видна четкая граница: до класса $(0.5; 0.7)$ включительно более эффективным (в смысле среднего времени решения задачи) является метод **Backward**, а после – метод **Forward**.

Было проведено большое количество экспериментов с различными значениями $m$ и $n$, в результате которых выяснилось, что граница эффективности рассматриваемых методов зависит не от самих значений $m$ и $n$, а от их соотношения.

Следует заметить, что не всегда существует точка переключения с метода **Backward** на метод **Forward**. В табл. 4, 5 приведены результаты экспериментов на тех классах, где возникает неопределенность в выборе точки переключения. Из табл. 4 видно, что при $m/n$, равном 0.23–0.32, на классе (0.5; 0.9) лучшим оказывается метод **Forward**, в то время как на следующем классе (0.7; 0.1) он проигрывает методу **Backward**; из табл. 5 видно, что при $m/n$, равном 0.61–0.75, на классах (0.1; 0.9) и (0.3; 0.3) лучшим оказывается метод **Forward**, в то время как на классах (0.3; 0.1) и (0.3; 0.5) лучшим является метод **Backward**.

Мы рекомендуем выбирать метод решения задачи в соответствии с табл. 6. В ней указаны точки переключения с метода **Backward** на метод **Forward** в линейном списке классов $(TF; RDD)$, указанном в табл. 3, в зависимости от $m/n$.

В заключение заметим, что поскольку метод **Backward** невыгодно использовать на классах с $TF = 0.9$, то можно рекомендовать использовать его в форме алгоритма 1-1 (см. обсуждение результатов эксперимента 1).

Работа выполнена при финансовой поддержке РФФИ (проект № 10-01-00728).

**Summary**

*I.K. Agapeevich, V.R. Fazylov.* Two Schemes of the Branch and Bound Method for a Flow Shop Total Weighted Tardiness Minimization Problem.

Two schemes of the branch and bound method for a flow shop total weighted tardiness minimization problem are suggested in this paper. The first scheme consists in the construction of the schedule in the common order (at the beginning the first work in the schedule is chosen, then the second work, etc.) and the second scheme presupposes the construction of the schedule in the inverse order (at the beginning the last work in the schedule is chosen, then the penultimate work, etc.). It is shown by numerical experiments that the efficiency of the methods strongly depends on the problem's parameters, which can be easily calculated from the initial data. Criteria for choosing the most efficient method for any particular problem are proposed.

**Key words:** flow shop, branch and bound method, total weighted tardiness minimization.

**Литература**

1. *Baker K.R.* Introduction to sequencing and scheduling. – N. Y.: Wiley, 1974. – 305 p.
2. *Bozejko W., Uchronski M., Wodecki M.* Scatter search for a weighted tardiness flow shop problem // Multidisciplinary Int. Scheduling Conf. MISTA 2009, Dublin, Ireland,

10–12 August 2009. – URL: http://staff.iiar.pwr.wroc.pl/wojciech.bozejko/papers/2009/MISTA_4pages.pdf, свободный.

3. *Bozejko W., Wodecki M.* Population-Based Heuristics for Hard Permutational Optimization Problems // Int. J. Comput. Intell. Res. – 2006. – V. 2, No 2. – P. 151–158.

4. *Bulbul K., Kaminsky Ph., Yano C.* Flow Shop Scheduling with Earliness, Tardiness, and Intermediate Inventory Holding Costs // Nav. Res. Logist. – 2004. – V. 51, No 3. – P. 407–445.

5. *Cicirello V.A.* On the Design of an Adaptive Simulated Annealing Algorithm // First Workshop on Autonomous Search, Rhode Island, USA, 23 September 2007. – URL: http://www.forevermar.com/SimA.pdf, свободный.

6. *Rahimi-Vahed A.R., Mirghorbani S.M.* A multi-objective particle swarm for a flow shop scheduling problem // J. Combin. Optim. – 2007. – V. 13, No 1. – P. 79–102.

7. *Sen T., Sulek J.M., Dileepan P.* Static scheduling research to minimize total weighted and unweighted tardiness: A state-of-the-art survey // Int. J. Prod. Econ. – 2003. – V. 3, No 1. – P. 1–12.

8. *Lenstra J.K., Rinnooy Kan A.H.G.* Complexity results for scheduling chains on a single machine // Eur. J. Oper. Res. – 1980. – V. 96. – P. 270–275.

9. *Engin O., Doyen A.* A new approach to solve flowshop scheduling problems by artificial immune systems // Future Gener. Comp. Sy. – 2004. – V. 20, No 6. – P. 1083–1095.

10. *Агапеевич И.К., Фазылов В.Р.* Генератор исходных данных для задачи минимизации суммарного взвешенного запаздывания в конвейерных системах // Исслед. по приклад. матем. и информат. – Казань: Изд-во Казан. ун-та, 2011. – Вып. 27. – С. 3–7.

11. *Конвей Р.В., Максвелл В.Л., Миллер Л.В.* Теория расписаний. – М.: Наука, 1975. – 359 с.

12. *Ignall E., Shrage L.* Application of the Branch-and-Bound Technique to Some Flow-Shop Scheduling Problems // Oper. Res. – 1965. – V. 13, No 3. – P. 400–412.

13. *Фазылов В.Р.* Задача манипулятора гальванической линии. – Казань: Казан. матем. о-во, 2000. – 79 с.

Поступила в редакцию
20.02.12

**Агапеевич Ильвина Константиновна** – соискатель кафедры экономической кибернетики Казанского (Приволжского) федерального университета.

E-mail: *AgapeevichIK@live.ru*

**Фазылов Валерий Рауфович** – доктор физико-математических наук, профессор кафедры экономической кибернетики Казанского (Приволжского) федерального университета.

E-mail: *vfazylov@gmail.com*