УДК 004: 519.854

**К. В. Кротов**

*Севастопольский государственный университет*
*ул. Университетская, 33, Севастополь, 299053, Россия*

*krotov_k1@mail.ru*

# ГРАДИЕНТНЫЙ МЕТОД ФОРМИРОВАНИЯ ДИНАМИЧЕСКИХ РАСПИСАНИЙ ОБРАБОТКИ ДАННЫХ В КОНВЕЙЕРНОЙ СИСТЕМЕ ПРИ РАЗЛИЧНЫХ МОМЕНТАХ ВРЕМЕНИ ИХ ПОСТУПЛЕНИЯ

Обосновываются модель составления расписаний обработки данных в конвейерной системе и метод построения динамических расписаний обработки данных при различных моментах времени их поступления в систему.

*Ключевые слова*: конвейерная система, расписания выполнения программ обработки данных, жадный алгоритм, динамические расписания, моменты времени поступления данных на обработку, локально эффективное решение, окрестности.

## Введение

Одним из способов повышения производительности выполнения программ обработки данных является конвейеризация, которая предполагает их разделение на фрагменты. Каждый фрагмент закреплен для выполнения за соответствующим сегментом конвейера. Введем в рассмотрение обозначения: $i$ – номер множества однотипных данных, характеризующих одинаковые объекты, которые обрабатываются в системе; $I$ – множество всех данных, которые обрабатываются в вычислительной системе; $n^i$ – количество элементов в множестве однотипных данных, характеризуемых индексом $i$. Индекс $i$ соответствует программе, выполняемой в составе конвейера, обрабатывающей данные $i$-го типа. Однократное выполнение конвейеризированной программы $i$-го типа обеспечивает обработку одного элемента множества данных $i$-го типа. Если множество данных $i$-го типа содержит $n^i$ элементов, то обрабатывающая эти данные программа должна быть выполнена в конвейерной системе $n^i$ число раз. При этом предполагается, что выполняющие обработку данных конвейеризированные программы находятся в оперативной памяти каждого из сегментов конвейера. Тогда управление вычислительным процессом в конвейерных системах предполагает определение порядка запуска программ обработки данных на выполнение. Так как объемы вычислений на каждом сегменте различны, различны и длительности выполнения программ на соответствующих сегментах, тогда может быть сформировано расписание выполнения конвейеризированных программ обработки соответствующих данных, представляющее собой порядок запуска программ на выполнение. Постановка задачи управления вычислительным процессом предполагает, что при $n^i = 1$ реализуется обработка единичных данных (однократный запуск на выполнение соответствующих программ), для которых должно быть сформировано расписание выполнения программ. В то же время ход вычислительного процесса подвержен возмущающим воздействиям следующих видов: поступление на обработку данных в различ-

*Кротов К. В.* Градиентный метод формирования динамических расписаний обработки данных в конвейерной системе при различных моментах времени их поступления // Вестн. Новосиб. гос. ун-та. Серия: Информационные технологии. 2016. Т. 14, № 1. С. 39–60.

ISSN 1818-7900. Вестник НГУ. Серия: Информационные технологии. 2016. Том 14, № 1

ные моменты времени, поступление на обработку данных в различные моменты времени и с разными приоритетами, отказы сегментов конвейера и т. д. Устранение влияния возмущений на ход вычислительного процесса (на выполнение сформированного статического расписания) реализуется путем построения динамических расписаний, учитывающих соответствующий вид возмущений.

**Анализ публикаций**

Современные методы теории расписаний позволяют формировать статические расписания обработки единичных данных разных типов при заданном количестве приборов в многостадийных обрабатывающих системах (в частности, в конвейерных системах) с использованием различных критериев. В работе [1] выполнен анализ основных подходов к решению задач формирования расписаний обработки данных. Для решения задач теории расписаний развиваются точные методы (ветвей и границ, ветвей и отсечений), приближенные методы и методы локальной оптимизации (метод отжига, генетические алгоритмы и т. д.). Реализация методов предполагает использование различных видов критериев оптимизации, учет ограничений на директивные сроки окончания обслуживания, возможность задания как фиксированного, так и произвольного маршрута обработки данных. Развитие методов теории расписаний связано с решением задач групповой обработки (обработки данных в партиях), однако методы построения расписаний при групповой обработке предполагают формирование фиксированных партий данных (включающих все данные одного типа) при обработке на ограниченном количестве приборов. Динамические свойства формируемых расписаний, связанные с различными событиями, происходящими в системе, ни в одной из анализируемых работ не рассматриваются. Таким образом, решение задачи построения динамических расписаний при учете возмущающих воздействий, изменяющих запланированный ход вычислительного процесса, является актуальным.

**Постановка цели научного исследования**

Цель работы состоит в совершенствовании методов построения расписаний обработки единичных данных в конвейерных системах. Совершенствование методов построения расписаний обработки единичных данных осуществляется с применением подхода дискретной оптимизации, связанного с поиском локально эффективных решений. Это позволит реализовать учет возмущающих воздействий на ход вычислительного процесса и реализовать построение динамических расписаний выполнения программ обработки данных в конвейерных системах. Возмущающим воздействием, учитываемым при формировании динамического расписания, является поступление в различные моменты времени данных в систему на обработку.

**Основное содержание работы**

Постановка задачи управления вычислительным процессом предполагает, что при $n^i = 1$ реализуется обработка единичных данных, для действий с которыми должно быть сформировано расписание выполнения программ. Если через $i$ обозначен идентификатор типа данных и идентификатор типа программы, выполняющей обработку этих данных, то через $d_i$ обозначен момент времени поступления в систему $i$-го типа данных. В рассматриваемой постановке задачи предполагается, что моменты времени поступления данных на обработку различны ($d_i \geq 0$). Обозначим через $l$ индекс сегмента вычислительной конвейерной системы, осуществляющего выполнение $l$-й части программы, при этом $l = \overline{1, L}$, где $L$ – общее количество сегментов конвейера. Каждым сегментом конвейерной системы выполняются вычисления, соответствующие назначенной для него части программы. Дисциплина обслуживания в системе предполагает прохождение данными всех сегментов конвейера, при этом если $l$-й сегмент приступил к выполнению $i$-й программы (к обработке данных $i$-го типа), обработка не может быть прервана обработкой других данных. Все обрабатывающие приборы конвейерной системы характеризуются равными и неизменными во времени значе-

ниями производительности их работы. Выполнение на каждом *l*-м сегменте назначенной ему части *i*-й программы характеризуется параметром длительности обработки данных на этом сегменте, однозначно соответствующей объему выполняемых вычислений при интерпретации программного кода. В системе выполняется обработка единичных данных, тогда $n^i = 1$, т. е. для каждой обрабатывающей программы будет выполнен ее однократный запуск на выполнение.

Особенностью решаемой задачи является задание двух групп обрабатываемых в системе данных. К первой группе, обозначенной $N_1$, относятся данных *n* типов, первоначально обрабатываемых в системе, т. е. данные, поступающие на обработку в момент времени $d_i = 0$ (данные, для которых первоначально формируется статическое расписание). Вторая группа, обозначенная как $N_2$, – данные, поступающие на обработку в отличные от 0 моменты времени, т. е. поступление которых на обработку реализуется после начала обработки данных первой группы (моменты времени поступления на обработку данных второй группы $d_i > 0$). Предполагается, что во вторую группу входит *k* типов данных. Тогда общее количество типов данных, обрабатываемых в системе (для которых $d_i = 0$ и $d_i > 0$), равно (*n* + *k*). Типы данных группы $N_2$ могут быть упорядочены в соответствии со значениями $d_i > 0$. Множество *N* обрабатываемых в системе данных различных типов определяется в виде $N = N_1 \cup N_2$, где множество $N_1$ содержит данные с $d_i = 0$, а множество $N_2$ – данные с $d_i > 0$. Решение задачи составления расписания осуществляется при условии, что параметры $d_i$ являются детерминированными.

Так как время обработки данных на сегментах конвейера различно, обработка данных более чем одного типа на каждом из сегментов невозможна, результаты обработки на предыдущем сегменте находятся в качестве исходных данных в оперативной памяти последующего сегмента, тогда управление вычислительным процессом состоит в определении порядка обработки данных на каждом из сегментов. Порядок обработки данных на сегментах должен быть определен таким образом, чтобы расписание обработки было эффективным в соответствии с формируемым критерием. Тогда каждому из сегментов соответствует некоторая последовательность $\pi^l$ обработки данных, а решение задачи – расписание обработки данных – определяется как совокупность последовательностей $\pi^l$, соответствующих *L* приборам в системе. Таким образом, расписание может быть представлено в виде $\pi = (\pi^1, \pi^2, ,..., \pi^L)$ ($l = \overline{1, L}$), последовательность $\pi^l$ ($l = \overline{1, L}$) имеет вид $\pi = (i_1^l, i_2^l, ,..., i_n^l, i_{n+1}^l, ..., i_{n+k}^l)$ и содержит как данные, поступившие на обработку в систему при $d_i = 0$ (*n* типов данных), так и данные с $d_i > 0$ (*k* типов данных). Для формализации видов последовательностей $\pi^l$ расписания $\pi$ в рассмотрение введены матрицы $(\boldsymbol{P})^l$ ($l = \overline{1, L}$) порядков обработки данных в системе (порядков запуска обрабатывающих программ на выполнение). Элемент $p_{ij}^l = 1$, если данные *i*-го типа занимают в последовательности $\pi^l$ *j*-ю позицию, $p_{ij}^l = 0$ в противном случае. Размерность введенных в рассмотрение матриц $(\boldsymbol{P})^l$ ($l = \overline{1, L}$) – $(n+k) \times (n+k)$.

Для формализации вида модели вычислительного процесса обработки данных (выполнения программ) в конвейерной системе в рассмотрение введены следующие обозначения: $t_i^l$ – длительность интервала обработки данных *i*-го типа на *l*-м сегменте конвейера ($l = \overline{1, L}$) соответствующей выполняющейся на нем программой; $(t_{ji}^{0l})$ – матрица моментов времени начала обработки данных *i*-го типа, занимающих в $\pi^l$ *j*-ю позицию. Через $\overline{t_{ji}^l}$ обозначим момент времени окончания обработки данных *i*-го типа, занимающих в $\pi^l$ *j*-ю позицию. Так как в системе выполняется обработка единичных данных, то длительности переналадки сегментов с обработки данных *i*-го типа на обработку данных *k*-го типа могут быть включены в интервалы $t_i^l$ обработки данных соответствующих *i*-х типов ($i = \overline{1, n+k}$) на *l*-х сегментах кон-

вейера ($l=\overline{1,L}$). Значения элементов матриц $(t_{ji}^{0l})$ $(l=\overline{1,L})$ определяются в соответствии с видом матриц $\left(\boldsymbol{P}\right)^l$ $(l=\overline{1,L})$ следующим образом: $t_{ji}^{0l}\neq 0$ для того элемента матрицы $(t_{ji}^{0l})$, который соответствует $p_{ij}^{l}=1$. В случае если $p_{ij}^{l}=0$, то соответствующий ему элемент $t_{ji}^{0l}$ матрицы $(t_{ji}^{0l})$ равен 0 ($t_{ji}^{0l}=0$). Для первого сегмента конвейера элементы матрицы $(t_{ji}^{01})$ определяются с учетом матрицы $\left(\boldsymbol{P}\right)^1$ следующим образом:

$$t_{1i}^{01}=0;\ t_{2i}^{01}=\sum_{h=1}^{n+k}t_h^1\cdot p_{h1}^1;\ t_{3i}^{01}=\sum_{j=1}^{2}\sum_{h=1}^{n+k}t_h^1\cdot p_{h,j}^1;\ t_{4i}^{01}=\sum_{j=1}^{3}\sum_{h=1}^{n+k}t_h^1\cdot p_{h,j}^1 \text{ и т. д.}$$

Понятно, что $t_{2i}^{01}\neq 0$ для того элемента матрицы $(t_{ji}^{01})$, который соответствует $p_{i2}^1=1$. В случае если $p_{i2}^1=0$, то соответствующий ему элемент матрицы $(t_{ji}^{01})$ равен 0. В общем виде выражения для определения $t_{ji}^{01}$ (номер позиции $j$ является заданным) имеют следующую форму:

$$t_{ji}^{0l}=\sum_{v=1}^{j-1}\sum_{h=1}^{n+k}t_h^1\cdot p_{h,v}^1\ \text{, при } j>1, \tag{1}$$

где $j$ – номер позиции данных $i$-го типа в последовательности $\pi^1$, для которых определяется значение $t_{ji}^{0l}$. Для $l$-го сегмента конвейера (при $l\neq 1$) элементы матрицы $(t_{ji}^{0l})$ определяются выражениями вида:

а) первая строка (первая позиция для данных $i$-го типа, $j=1$) –

$$t_{1,i}^{0l}=\sum_{h=1}^{n+k}p_{ih}^{l-1}\cdot(t_{h,i}^{0l-1}+t_i^{l-1}), \tag{2}$$

где индекс $i$ типа данных определяется по матрице $\left(\boldsymbol{P}\right)^l$ $(l=\overline{1,L})$ как занимающих первую позицию в последовательности $\pi^l$ ($i$-й тип данных, обрабатываемых в позиции $j=1$ на $l$-м сегменте конвейера);

б) $j$-я строка ($j\neq 1$) в матрице $\left(\boldsymbol{t_{ji}^{0l}}\right)$ $(l=\overline{2,L})$ для данных $i$-го типа –

$$t_{ji}^{0l}=max\left\{\sum_{h=1}^{n+k}p_{i,h}^{l-1}\cdot(t_{h,i}^{0l-1}+t_i^{l-1});\sum_{h=1}^{n+k}(t_{j-1,h}^{0l}+t_h^l)\cdot p_{h,j-1}^l\right\}. \tag{3}$$

Выражения (1)–(3) являются моделью вычислительного процесса обработки данных конвейеризированными программами в многостадийной системе, т. е. позволяют определять его временные характеристики. Формируемые значения временных характеристик вычислительного процесса зависят от видов матриц $\left(\boldsymbol{P}\right)^l$ $(l=\overline{1,L})$. Тогда сформированное решение может быть представлено в форме $[(\boldsymbol{P})^l,(t_{ji}^{0l})\,|\,l=\overline{1,L}]$. Для анализа эффективности получаемых решений должен быть сформирован критерий, учитывающий рассчитываемые с использованием выражений (1)–(3) значения характеристик вычислительного процесса.

Особенностями алгоритма метода определения порядка обработки данных, используемого при построении расписания выполнения программ и реализующего «жадную» стратегию [1], являются: 1) добавление в конец сформированных на предыдущем ($(s-1)$-м) шаге алгоритма последовательностей выполнения программ $\pi^l(s-1)$ $(l=\overline{1,L})$ программы $i$-го типа для обработки соответствующих ей данных; 2) определение эффективного местоположения (позиции) рассматриваемой программы для $i$-го типа данных во вновь формируемых последовательностях на текущем ($s$-м) шаге алгоритма $\pi^l(s)$; 3) вычисление и анализ градиентов целевой функции после реализации каждого шага алгоритма, связанного с изменением положения в одной последовательности $\pi^l$ рассматриваемых данных $i$-го типа на одну позицию ближе к ее началу; 4) в случае если изменение положения данных рассматриваемого $i$-го типа в последовательностях $\pi^l$ $(l=\overline{1,L})$ на одну позицию ближе к их началу не приводит

к уменьшению значения целевой функции, тогда реализуется изменение положения данных *i*-го типа одновременно в двух последовательностях $\pi^l$ ($l=\overline{1,L}$), далее в трех последовательностях и т. д.; таким образом, для текущего решения реализуется переход к более эффективному решению в рамках его окрестностей с различными метриками (максимальная метрика окрестности является заданной).

В соответствии с особенностями алгоритма формирования расписаний обработки данных на некотором *s*-м шаге выполняется решение отдельной подзадачи добавления и размещения в последовательностях $\pi^l$ ($l=\overline{1,L}$) данных одного *i*-го типа, при этом на последующих шагах алгоритма предполагается решение подзадачи размещения в последовательностях $\pi^l$ ($l=\overline{1,L}$) данных оставшихся ($n-i$)-го типов. Таким образом, на *s-м* шаге алгоритма выполняется жадный выбор по определению локально эффективного решения (по определению эффективных *j*-х позиций в $\pi^l$ ($l=\overline{1,L}$) данных рассматриваемого *i*-го типа). На основе локально эффективного решения для данных *i*-го типа (представляющего собой порядок обработки данных всех типов, добавленных в $\pi^l$ на предшествующих шагах алгоритма) формируется решение по определению позиций данных последующих ($n-i$)-го типов, при этом порядок обработки данных, полученный на предшествующих шагах алгоритма, не меняется. В итоге каждый последующий жадный выбор связан с добавлением в $\pi^l$ данных одного типа, определением их позиции в этих последовательностях; при этом решение по размещению данных рассматриваемого *i*-го типа формируется на основе локально эффективных видов последовательностей, сформированных для данных предшествующих ($i-1$)-го типа, и порядок обработки данных этих типов в $\pi^l$ не изменяется.

При учете того, что на каждом *s*-м шаге алгоритма в полученные ранее локально эффективные последовательности $\pi^l$ добавляется одна программа обработки данных *i*-го типа, возможна оценка эффективности видов последовательностей для текущего количества программ в них. Иначе говоря, не все программы одновременно находятся в последовательностях $\pi^l(s)$ ($l=\overline{1,L}$), а только их некоторое текущее количество, последовательность которых оценивается. Для обоснования вида критерия эффективности формируемого решения выполнены следующие рассуждения.

1. При $\sum\limits_{i=1}^{n+k} t_{ji}^{0l}\cdot p_{ij}^{l} > \sum\limits_{i=1}^{n+k}(t_{j-1,i}^{0l}+t_i^l)\cdot p_{i,j-1}^{l}$ *l*-й сегмент конвейера ожидает готовности для обработки данных в *j*-й позиции после их обработки в ($j-1$)-й позиции (здесь рассматриваются позиции данных, а не их тип); длительность простоев *l*-го сегмента конвейера в ожидании данных, занимающих *j*-ю позицию в последовательности $\pi^l$, определяется следующим образом: $\sum\limits_{i=1}^{n+k} t_{j,i}^{0l}\cdot p_{i,j}^{l}-\sum\limits_{i=1}^{n+k}(t_{j-1,i}^{0l}+t_i^l)\cdot p_{i,j-1}^{l}$, длительность простоев *l*-го сегмента при выполнении текущего количества программ, находящихся в $\pi^l$ ($l=\overline{1,L}$), определяется выражением $\sum\limits_{j=2}^{n+k}(\sum\limits_{i=1}^{n+k} t_{j,i}^{0l}\cdot p_{i,j}^{l}-\sum\limits_{i=1}^{n+k}(t_{j-1,i}^{0l}+t_i^l)\cdot p_{i,j-1}^{l})$; полученное выражение позволяет определять суммарные простои *l*-го сегмента в ожидании готовности данных при их обработке в позициях с ($j=2$) по $n+k$ (простои после начала обработки данных в $\pi^l$ как для статического, так и для динамического расписаний).

2. Для последовательности $\pi^1$ простои первого сегмента конвейера в ожидании готовности данных отсутствуют, тогда суммарное время простоев всех ($L-1$)-го сегмента конвейерной системы, связанных с ожиданием готовности данных для обработки, будет определено выражением вида: $\sum\limits_{l=2}^{L}\sum\limits_{j=2}^{n+k}\left(\sum\limits_{i=1}^{n+k} t_{j,i}^{0l}\cdot p_{i,j}^{l}-\sum\limits_{i=1}^{n+k}\left(t_{j-1,i}^{0l}+t_i^l\right)\cdot p_{i,j-1}^{l}\right)$.

3. Простои сегментов конвейера при выполнении программ связаны также с ожиданием поступления данных на обработку, если эти данные занимают первую позицию в последова-

тельностях $\pi^l$ ($l=\overline{2,L}$); для $l$-го сегмента (при условии, что $l \neq 1$) время ожидания готовности данных для обработки в первой позиции $\pi^l$ ($l=\overline{2,L}$) определяется выражением вида $\sum_{i=1}^{n+k} t_{1,i}^{0l} \cdot p_{i1}^{l}$; так как $t_{1i}^{01}=0$, простои всех $l$-х сегментов конвейера ($l=\overline{2,L}$) будут определены исходя из выражения $\sum_{l=2}^{L}\sum_{i=1}^{n+k} t_{1,i}^{0l} \cdot p_{i1}^{l}$.

В соответствии с выполненными рассуждениями конечный вид выражения для критерия эффективности формируемых решений по порядку выполнения программ (порядку обработки данных) в последовательностях $\pi^1$ ($l=\overline{1,L}$) для статического и динамического расписаний следующий:

$$f=\sum_{l=2}^{L}\sum_{i=1}^{n+k} t_{1,i}^{0l} \cdot p_{i,1}^{l}+\sum_{l=2}^{L}\sum_{j=2}^{n+k}\Big(\sum_{i=1}^{n+k} t_{ji}^{0l} \cdot p_{ij}^{l}-\sum_{i=1}^{n+k}\big(t_{j-1,i}^{0l}+t_{i}^{l}\big)\cdot p_{i,j-1}^{l}\Big). \qquad (4)$$

Введем обозначения: $O$ – окрестность текущего рассматриваемого решения $\pi(s)$, в которой выполняется поиск нового локально эффективного решения; $O_1$ – окрестность с метрикой 1; $\pi^l$ – окрестность с метрикой $h$; $O_{h_{\max}}$ – окрестность с максимальной рассматриваемой метрикой $h_{\max}$ (максимально возможная метрика окрестности рассматриваемого решения). Значение метрики $h$ окрестности $O_h$ ($h=\overline{1,h_{\max}}$) определяется выражением вида $h=\sum_{i=1}^{n+k}\sum_{j=1}^{n+k}|\,p_{i,j}(s+v)-p_{i,j}(s)\,|/2$, где $v$ – индекс промежуточного шага алгоритма, на котором выполняется формирование нового решения $\pi(s+v)$ на базе локально эффективного решения $\pi(s)$. При $h=1$ положение данных рассматриваемого $i$-го типа изменяется в одной из последовательностей $\pi^l$ ($l=\overline{1,L}$) ближе к их началу, при $h=2$ положение данных $i$-го типа изменяется в двух последовательностях $\pi^l$ ($l=\overline{1,L}$) и т. д.

Алгоритм определения эффективного положения обрабатываемых данных $i$-го типа в последовательностях $\pi^l$ ($l=\overline{1,L}$) (при построении статических расписаний) предполагает: 1) размещение данных рассматриваемого $i$-го типа в $\pi^l$ ($l=\overline{1,L}$) путем добавления их в конец каждой последовательности; 2) изменение положения данных $i$-го типа в последовательностях $\pi^l$ ($l=\overline{1,L}$) на одну позицию ближе к началу последовательностей в зависимости от текущего значения метрики $h$ окрестности $O_h(s)$ (в одной последовательности $\pi^l$ ($l=\overline{1,L}$) – метрика $h=1$, в двух последовательностях $\pi^l$ – метрика $h=2$, в трех последовательностях – метрика $h=3$ и т. д.); 3) изменение положения данных $i$-го типа в последовательностях $\pi^l$ обуславливает изменение значения критерия $f$ вида (4), при этом направление дискретного градиента критерия $f$, обозначенное как $v$, сопоставляется с идентификаторами последовательностей $\pi^l$ (с группой последовательностей $\pi^l$), в которых изменяется положение рассматриваемых данных (рассматриваются левый ${}^{-}\nabla f_v \le 0$ либо правый ${}^{+}\nabla f_v > 0$ дискретные градиенты целевой функции $f$ [2]), при ${}^{-}\nabla f_v\,(\pi(s)) \le 0$ – левый дискретный градиент целевой функции $f$, соответствующий расписанию $\pi(s+v)$, сформированному на основе исходного расписания $\pi(s)$, для которого положение данных $i$-го типа зафиксировано как текущее локально эффективное на $s$-м шаге; 4) в случае если гарантируется выполнение условия ${}^{-}\nabla f_v \le 0$ вдоль направлений $v$, тогда среди всех $v$, для которых выполняется это условие, выбирается то направление $v'$, для которого $\max_v\big(\big|{}^{-}\nabla_v f(\pi(s)) \le 0\big|\big)$; 5) в тех последовательностях $\pi^l$, в которых изменение положений данных $i$-го типа гарантирует выполнение условия $\max_v\big(\big|{}^{-}\nabla_v f(\pi(s)) \le 0\big|\big)$, позиции данных фиксируются (т. е. рас-

сматриваемое решение фиксируется как локально эффективное), после чего действия по определению нового локально эффективного положения данных *i*-го типа в последовательностях $\pi^l$ ($l=\overline{1,L}$) повторяются; 6) в том случае, если при поиске локально эффективного решения в окрестности $O_h(s)$ с меньшей метрикой такое решение не найдено, то значение метрики увеличивается на 1, после чего положение рассматриваемых данных *i*-го типа изменяется на одну позицию ближе к началу последовательностей $\pi^l$ ($l=\overline{1,L}$), количество которых соответствует значению метрики (рассматриваемые действия повторяются до тех пор, пока значение метрики окрестности для формируемых решений не превысит максимально возможное значение $h_{\max}$, увеличение метрики окрестности выполняется до тех пор, пока $h \le h_{\max}$). Таким образом, направление *v* дискретного градиента целевой функции *f* соответствует идентификаторам последовательностей $\pi^l$, в которых изменяется положение данных рассматриваемого *i*-го типа на (*s* + *v*)-м шаге алгоритма, т. е. направление *v* соответствует индексу шага *v* определения эффективного решения на основе текущего локально эффективного решения $\pi(s)$, а также некоторой *v*-й группе последовательностей $\pi^l$, в которых на данном шаге алгоритма одновременно изменяется положение данных рассматриваемого *i*-го типа. Формулировка алгоритма метода построения статических расписаний обработки данных, основывающегося на жадных стратегиях, изложена в [1]. Этот метод используется при построении статического расписания для данных с $d_i = 0$.

Учет влияния возмущающих воздействий на запланированный ход вычислительного процесса возможен путем построения динамических расписаний. Запланированный ход вычислительного процесса реализуется в соответствии со сформированным статическим расписанием (для этого используется градиентный метод, основывающийся на жадных стратегиях [1]). Формирование динамических расписаний предполагает перестроение полученного эффективного статического расписания обработки (для данных с $d_i = 0$) с учетом момента времени $d_i > 0$ поступления на обработку в конвейерную систему данных, входящих в группу $N_2$. По этой причине должны быть обоснованы особенности формирования последовательностей $\pi^l$ ($l=\overline{1,L}$) при построении динамических расписаний с учетом поступления данных *k* типов, входящих в группу $N_2$. При построении статических и динамических расписаний используется один и тот же вид критерия эффективности.

Предложенный в [1] метод построения статических расписаний обработки данных в конвейерных системах является основой для обоснования метода построения динамических расписаний, учитывающего моменты времени $d_i > 0$ поступления данных в конвейерную систему. При этом первоначально реализуется формирование статического расписания обработки данных, которое модифицируется с использованием формулируемого метода (после поступления данных в систему на обработку в момент времени $d_i > 0$ полученное статическое расписание модифицируется с использованием метода построения динамических расписаний).

Для реализации дальнейших рассуждений выполнен переход к обозначениям типа данных и их позиций в последовательности $\pi^l$ ($l=\overline{1,L}$) в виде $i_j^l$, где *j* – позиция данных *i*-го типа в рассматриваемой последовательности $\pi^l$ *l*-го сегмента. Данные, входящие в группу $N_2$ (для которых $d_2 > 0$), должны быть добавлены в уже сформированные последовательности $\pi^l$ ($l=\overline{1,L}$) таким образом, чтобы в результате было получено эффективное расписание выполнения программ обработки данных как множества $N_1$, так и множества $N_2$. Добавление данных множества $N_2$ к уже сформированным для данных множества $N_1$ последовательностям $\pi^l$ ($l=\overline{1,L}$) позволяет получить новые расписания $\pi$ для множества типов данных $N = N_1 \cup N_2$.

В рассмотрение введено множество $N_2^{'}$, сформированное на основе множества $N_2$ таким образом, чтобы данные *i*-х типов ($i \in N_2$) были упорядочены в соответствии со значением $d_i > 0$, т. е.

$$N_2^{'} = \{i_{n+1}, i_{n+2}, ..., i_{n+k} \mid i_{n+g} \in N_2, d_{n+g} > 0, d_{n+(g-1)} \le d_{n+g}, g = \overline{1,k}\}.$$

Обозначение $i_{n+g} \in N'_2$ ($g = \overline{1,k}$) соответствует *i*-му типу данных с $(n+g)$-м порядковым номером в множестве $N_2^{'}$. Элементы множества $N_2^{'}$ – это данные, размещение которых в последовательностях $\pi^l$ ($l = \overline{1,L}$) должно быть реализовано эффективным образом. Выполним дальнейшие рассуждения для одного из типов данных $i_{n+g} \in N'_2$, которые должны быть добавлены в последовательности $\pi^l$ ($l = \overline{1,L}$). Аналогичные действия выполняются для любого типа данных $i_{n+g} \in N'_2$ ($g = \overline{1,k}$).

Если последовательности данных $\pi^l(s-1)$ для расписания $\pi(s-1)$, сформированного на предыдущем $(s-1)$-м шаге алгоритма, имеют вид $\pi^l(s-1) = \{i_1^l, i_2^l, ..., i_{n+(g-1)}^l\}$, тогда вид последовательностей $\pi^l(s)$ с добавленными в них данными $i_{n+g} \in N'_2$ следующий:

$$\pi^l(s) = \pi^l(s-1) \bigcup \{i_{n+g}\}, \text{ где } \pi^l(s-1) = \{i_1^l, i_2^l, ..., i_{n+(g-1)}^l\},$$

$$\pi^l(s) = \{i_1^l, i_2^l, ..., i_{n+(g-1)}^l, i_{n+g}^l\}. \quad (5)$$

Выражение (5) соответствует начальному виду последовательностей $\pi^l(s)$ ($l = \overline{1,L}$), в которых в ходе реализации алгоритма определяется эффективное местоположение (позиция) данных $i_{n+g}$. К моменту времени $d_{i_{n+g}} > 0$ ($i_{n+g} \in N'_2$) некоторые данные $i_j^l$ в последовательностях $\pi^l(s-1)$ могут быть обработаны либо находиться в процессе обработки. Тогда в последовательности $\pi^l(s-1)$ могут быть выделены две подпоследовательности $\pi_1^l(s-1)$ и $\pi_2^l(s-1)$ в соответствии с условиями

$$\pi_1^l(s-1) = \{i_1^l, i_2^l, ..., i_q^l \mid t_{i_j}^{0l} \le d_{i_{n+g}}, j = \overline{1,q}, g = \overline{1,k}\}, \quad (6)$$

$$\pi_2^l(s-1) = \{i_{q+1}^l, i_{q+2}^l, ..., i_{n+(g-1)}^l \mid t_{i_j}^{0l} > d_{i_{n+g}}, j = \overline{q+1, n+(g-1)}, g = \overline{1,k}\}. \quad (7)$$

Так как $i_{n+g}$ – это идентификатор (индекс) данных, добавляемых в последовательности $\pi^l(s)$ ($l = \overline{1,L}$) на текущем *s*-м шаге алгоритма (первоначально размещаемых в конце последовательностей $\pi^l(s)$ (последовательности $\pi_2^l(s)$, $l = \overline{1,L}$)), модифицированный вид последовательности $\pi_2^l(s)$ ($l = \overline{1,L}$) следующий:

$$\pi_2^l(s) = \{i_{q+1}^l, i_{q+2}^l, ..., i_{n+(g-1)}^l, i_{n+g}^l \mid t_{i_j}^{0l} > d_{i_{n+g}}, j = \overline{q+1, n+(g-1)}, g = \overline{1,k}\}. \quad (8)$$

Таким образом, последовательность $\pi^l(s)$, эффективный порядок обработки данных в которой определяется на *s*-м шаге алгоритма, может быть представлена в виде

$$\pi^l(s) = \pi_1^l(s) \bigcup \pi_2^l(s).$$

Условие (6) позволяет определить последовательность данных, порядок которых изменен быть не может, так как момент времени начала их обработки предшествует моменту времени поступления на обработку данных $i_{n+g} \in N'_2$ ($g = \overline{1,k}$). В последовательности $\pi_1^l(s)$ ($l = \overline{1,L}$) данные $i_q^l$ – ее правая граница, до которой может изменяться позиция (местоположение) данных $i_{n+g}$ ($g = \overline{1,k}$) в последовательности $\pi_2^l(s)$. Условие (7) позволяет определить в каждой из последовательностей $\pi^l(s)$ те данные, порядок которых может быть изменен с учетом поступивших в момент времени $d_{i_{n+g}} > 0$ на обработку данных $i_{n+g} \in N'_2$ ($g = \overline{1,k}$). Таким обра-

зом, рассматриваемые на текущем *s*-м шаге алгоритма данные $i_{n+g} \in N'_2$ (с $d_{i_{n+g}} > 0$) могут занимать в последовательностях $\pi_2^l(s)$, определенных в соответствии с условием (7), любую из позиций (если это гарантирует улучшение значения целевой функции (4)) до данных $i_q^l$, являющихся правой границей последовательности $\pi_1^l(s)$ $(l = \overline{1,L})$.

Исходной информацией для решения задачи введения данных множества $N'_2$ в последовательности обработки $\pi^l(s)$ $(l = \overline{1,L})$ на *s*-м шаге алгоритма являются: 1) тип данных $i_{n+g} \in N'_2$, которые добавляются в последовательности $\pi^l(s)$ $(l = \overline{1,L})$ и эффективное местоположение которых в этих последовательностях определяется; 2) расписание $\pi(s-1)$, сформированное на предыдущем (*s* – *1*)-м шаге алгоритма, состоящее из последовательностей $\pi^l(s-1) = \{i_1^l, i_2^l, ..., i_{n+(g-1)}^l\}$ $(l = \overline{1,L})$, в каждой последовательности $\pi^l(s-1)$ $(l = \overline{1,L})$ которого в соответствии с условиями (6) и (7) определяются подпоследовательности $\pi_1^l(s)$ и $\pi_2^l(s)$. Таким образом, на текущем *s*-м шаге алгоритма в последовательности $\pi^l(s-1)$ $(l = \overline{1,L})$ добавляются данные ($i_{n+g}$)-го типа и определяется эффективное местоположение их в этих последовательностях.

Сформулированные условия (6), (7) позволяют определить состав последовательностей $\pi_1^l$ и $\pi_2^l$ $(l = \overline{1,L})$ в расписании, построенном для данных $n + (g-1)$ типов (для данных с $d_i \geq 0$, включенных в последовательности $\pi^l$ до текущего *s*-го шага). Таким образом, количество данных в последовательностях $\pi^l(s-1)$, состоящих из подпоследовательностей $\pi_1^l$ и $\pi_2^l$ $(l = \overline{1,L})$, равно $n + (g-1)$.

Введем в рассмотрение множества $N^l$ $(l = \overline{1,L})$ данных (типов данных) в последовательностях $\pi_2^l$, порядок которых при построении динамического расписания может быть изменен. На основе условия (7) и сформированного вида последовательностей $\pi_2^l$ для каждого *l*-го сегмента определено множество $N^l$ $(l = \overline{1,L})$ тех данных, порядок которых в последовательностях $\pi_2^l$ будет определяться. Формируемые с учетом (7) и видов последовательностей $\pi_2^l$ множества $N^l$ для *l*-х сегментов $(l = \overline{1,L})$ имеют вид $N^l = \{i_{q+1}^l, i_{q+2}^l, ..., i_{n+(g-1)}^l\}$. В каждое множество $N^l$ должны быть добавлены данные ($i_{n+g}$)-го типа, размещаемые в $\pi^l$ на *s*-м (текущем) шаге алгоритма, т. е. $N^l = N^l \cup \{i_{n+g}\}$ $(l = \overline{1,L})$. Для данных, которые не были обработаны к моменту времени $d_{i_{n+g}} > 0$ поступления на обработку данных $i_{n+g}$, в рассмотрение введено множество $N_{np}$ (not processed), которое формируется следующим образом:

$$N_{np} = \bigcup_{l=1}^{L} N^l .$$

Для определения порядка действий алгоритма составления динамических расписаний необходимо предварительно ввести в рассмотрение способ формирования последовательностей $\pi_2^l$ $(l = \overline{1,L})$ и сформулировать способ определения их эффективного вида. Расчет значений критерия (4) возможен в случае, когда количество данных в последовательностях $\pi^l$ $(l = \overline{1,L})$ одинаково, при этом в $\pi^l$ размещены данные одинаковых типов. Если проинтерпретировать $\pi^l$ $(l = \overline{1,L})$ как упорядоченные множества данных, тогда для расчета значений критерия (4) необходимо, чтобы $|\pi^l| = |\pi^{l'}|$, где $l$ и $l'$– индексы различных последовательностей $\pi^l$ и $\pi^{l'}$ $(l \neq l')$. Поэтому способ формирования последовательностей $\pi_2^l$ $(l = \overline{1,L})$ предусматривает реализацию двух последовательных этапов:

1) размещение в последовательностях $\pi_2^l$ ($l=\overline{1,L}$) всех данных $i \in N_{np}$ таких, что $i \notin \pi_1^l$ и $i \in \pi_1^{l'}$, где $l \neq l'$(данные $i'$ не входят в последовательность $\pi_1^l$ $l$-го сегмента конвейера, но входят в последовательность $\pi_1^{l'}$ другого $l'$-го сегмента, при $l \neq l'$), определение в $\pi_2^l$ эффективного положения этих данных; таким образом, на первом этапе определяется эффективное местоположение в $\pi_2^l$ (в итоге – в $\pi^l$) всех данных $i$, для которых перед началом построения динамического расписания выполняется условие $i \in N_{np} \cap (\bigcup_{l=1}^{L} \pi_1^l)$;

2) размещение в последовательностях $\pi_2^l$ данных $i$-х типов, для которых перед началом построения динамического расписания выполняется условие $(i \in N_{np}) \& (i \notin (\bigcup_{l=1}^{L} \pi_1^l))$, и определение эффективного местоположения этих данных в последовательностях $\pi_2^l$ (т. е. размещение в последовательностях $\pi_2^l$ данных $i$-х типов, которые перед началом построения расписания не входили в последовательности $\pi_1^l$ ни на одном $l$-м сегменте ($l=\overline{1,L}$)) ; в итоге на втором этапе в последовательностях $\pi_2^l$ размещаются (определяется их эффективное местоположение) все данные, для которых выполняется условие $(i \in N_{np}) \& (i \notin (\bigcup_{l=1}^{L} \pi_1^l))$, в том числе рассматриваемые данные $i_{n+g}$.

Для реализации первого этапа формирования последовательностей $\pi_2^l$ необходимо: а) определить все данные $i$, для которых выполняется условие $i \in N_{np} \cap (\bigcup_{l=1}^{L} \pi_1^l)$ (которые будут добавлены в $\pi_2^l$ на первом этапе построения динамического расписания), для этих данных сформировать некоторое начальное решение по порядку их обработки в последовательностях $\pi_2^l$ ($l=\overline{1,L}$); б) сформировать эффективные порядки обработки данных, добавленных в $\pi_2^l$, с учетом последовательностей $\pi_1^l$ ($l=\overline{1,L}$). Иначе говоря, на основе сформированного начального решения при учете порядков данных в последовательностях $\pi_1^l$ ($l=\overline{1,L}$) выполнить определение эффективных позиций $j$ данных $i$-х типов ($i \in N_{np} \cap (\bigcup_{l=1}^{L} \pi_1^l)$). В рассмотрение введено множество $N'_{np}$ типов данных $i$ таких, что $i \in N_{np} \cap (\bigcup_{l=1}^{L} \pi_1^l)$; если для некоторого типа данных $i \in N_{np}$ это условие выполняется, то $N'_{np} = N'_{np} \cup \{i\}$. Формирование начального решения по порядкам обработки данных в последовательностях $\pi_2^l$ ($l=\overline{1,L}$) предполагает добавление в них данных $i$, для которых выполняются условия: $i \in N'_{np}$ и $i \notin \pi_1^l$ ($l=\overline{1,L}$) (в $\pi_2^l$ на $l$-м сегменте добавляются данные $i$-го типа, которые не входят в последовательность $\pi_1^l$ на этом сегменте, но содержатся в последовательности $\pi_1^{l'}$ другого $l'$-го сегмента ($l \neq l'$)). В силу выполненных рассуждений порядок шагов алгоритма, реализующего формирование начального решения по составам последовательностей $\pi_2^l$ ($l=\overline{1,L}$) для данных $i \in N'_{np}$, имеет следующий вид (множество $N'_{np}$ предварительно упорядочивается по возрастанию идентификаторов $i$ типов данных в нем):

1) из множества $N'_{np}$ выбирается тип данных $i'$ в соответствии с условием $i' = \min\{i \mid i \in N'_{np}\}$ (определяются данных $i'$-го типа, добавляемые в последовательности $\pi_2^l$);

2) для данных $i'$-го типа и каждой последовательности $i' \in \pi^l$ ($l = \overline{1,L}$) выполняется проверка условия $i' \in \pi_1^l$, в случае выполнения этого условия данные $i'$-го типа не могут быть добавлены в $\pi_2^l$; в случае $i' \notin \pi_1^l$ данные $i'$ добавляются в конец последовательности $\pi_2^l$ ($\pi_2^l = \pi_2^l \cup \{i'\}$);

3) после добавления типа данных $i'$ в последовательности $\pi_2^l$ выполняется его удаление из множеств $N'_{пр}$ и $N_{пр}$: $N'_{пр} = N'_{пр} \setminus \{i'\}$; $N_{пр} = N_{пр} \setminus \{i'\}$ (данные $i'$ размещены в $\pi_2^l$, поэтому рассматриваться в качестве элементов множеств $N_{пр}$ и $N'_{пр}$ не будут); проверка условия $N'_{пр} \neq \varnothing$; в случае выполнения условия реализуется переход на шаг 1; при невыполнении условия все данные *i*-х типов ($i \in N'_{пр}$) размещены в $\pi_2^l$, останов алгоритма.

В результате для *l*-х сегментов конвейера ($l = \overline{1,L}$) сформированы последовательности $\pi_2^l$ одного из следующих видов: 1) $\pi_2^l = \varnothing$, если ни один из типов данных $i \in N'_{пр}$ в последовательность $\pi_2^l$ добавлен не был; 2) $\pi_2^l = \{i_{q+1}^l, i_{q+2}^l, ..., i_{q+n_l}^l\}$, где $n_l$ – количество данных, которые были добавлены в последовательность $\pi_2^l$. Во втором случае $(q+1)$ – номер позиции в $\pi^l$ данных, являющихся первыми в последовательности $\pi_2^l$ (начальная $(q+1)$ – я позиция данных в $\pi_2^l$ выступает следующей за крайней правой *q*-й позицией данных в последовательности $\pi_1^l$).

На основе начального решения для данных $i \in N'_{пр}$ в последовательностях $\pi_2^l$ ($l = \overline{1,L}$) необходимо реализовать определение эффективных порядков обработки данных в этих последовательностях. При этом следует учесть: последовательности $\pi_2^l$ содержат данные в порядке возрастания значений идентификатора $i$ их типа; после формирования начального решения для последовательностей $\pi_2^l$ количество данных в $\pi^l$ ($l = \overline{1,L}$) одинаково. Для определения эффективных порядков обработки данных в последовательностях $\pi_2^l$ на основе сформированного начального решения (при условии рассмотрения последовательностей $\pi_2^l$ как упорядоченных множеств элементов) выполним повторную инициализацию множества $N'_{пр}$ следующим образом: $N'_{пр} = \bigcup_{l=1}^{L} \pi_2^l$, тогда множество $N'_{пр}$ – это типы данных, порядок которых в последовательностях $\pi_2^l$ будет определяться. Полученное множество $N'_{пр}$ упорядочивается по возрастанию идентификаторов типов данных $i \in N'_{пр}$.

Для определения эффективных порядков обработки данных в последовательностях $\pi_2^l$ на основе начального решения использован подход к изменению позиций данных, предложенный в [3–6]. В соответствии с [3–6] рассматривается некоторая текущая последовательность $\pi_2^l$, в которой на предварительном этапе размещено $n_l$ данных и реализуется перемещение в ней единичных данных (данных одного типа) из текущей позиции на новое местоположение (изменение местоположения единичных данных в $\pi_2^l$), либо взаимная перестановка двух различных единичных данных в этой последовательности.

Через $i'$ обозначим тип данных в последовательностях $\pi_2^l$, позиция которых будет изменяться на текущем *s*-м шаге алгоритма формирования эффективных порядков обработки данных ($i'$ – рассматриваемый тип данных, эффективное местоположение которых в последовательностях $\pi_2^l$ определяется на текущем шаге алгоритма); $j^l$ – текущая позиция данных рассматриваемого типа $i'$ в последовательности $\pi_2^l$ ($l = \overline{1,L}$). Тогда данные $i'$-го типа занимают в последовательности $\pi_2^l$ ($l = \overline{1,L}$) начальную позицию $j^l$, которая в процессе поиска эффективного решения будет изменяться. В результате для некоторого исходного начального

решения $\pi$ (порядков обработки данных в последовательностях $\pi^l$) определяется окрестность $O_1(\pi)$, состоящая из решений, отличающихся от исходного решения $\pi$ тем, что в каждой из последовательности $\pi_2^l$ данные $i'$-го типа (если они входят в эти последовательности) перемещаются на одну позицию ближе к началу (выполняется изменение позиции $j^l$ данных $i'$-го типа в $\pi_2^l$). При одновременном изменении положения данных одного типа сразу в нескольких последовательностях $\pi_2^l$ (по аналогии с [1]) может быть выполнен переход от окрестности $O_1(\pi)$ к окрестности $O_h(\pi)$, при $h=\overline{1,h_{\max}}$ ($h_{\max}$ – максимальное количество последовательностей $\pi_2^l$, в которых одновременно может быть изменена позиция данных рассматриваемого $i'$-го типа, $h_{\max}$ – метрику максимальной окрестности $O_h(\pi)$). По аналогии с [1] на *s-м* шаге алгоритма рассматриваемые данные $i'$ перемещаются на одну позицию к началу в каждой последовательности $\pi_2^l$ (при $i'\in\pi_2^l$) либо в совокупности последовательностей $\pi_2^l$, количество которых определяется индексом $h$ ($h=\overline{1,h_{\max}}$). В формируемых окрестностях $O_h(\pi)$ ($h=\overline{1,h_{\max}}$) реализуется определение более эффективных решений, чем текущее решение $\pi$. Затем для полученного нового локально эффективного решения рассматриваются окрестности $O_h(\pi)$ ($h=\overline{1,h_{\max}}$), состоящие из сформированных на его основе расписаний обработки данных, построенных в соответствии с описанным выше подходом. Если в сформированных окрестностях $O_h(\pi)$ более эффективное решение не найдено, реализуется переход к следующему типу данных $i'\in N'_{пр}$, исходное местоположение (позиция) $j^l$ которых в последовательностях $\pi_2^l$ будет изменяться.

В соответствии с выполненными рассуждениями сформулирован алгоритм определения эффективных порядков обработки данных, размещенных в $\pi_2^l$ на первом этапе их формирования. Расписание $\pi(s)$ характеризуется совокупностью матриц $P^l(s)$, $(t_{ji}^{0l}(s))$ ($l=\overline{1,L}$), поэтому начальное решение как локально эффективное обозначим как ($P^l(s)$; $(t_{ji}^{0l}(s))\mid l=\overline{1,L}$)*. Введем обозначение $N_s^p$ для множества направлений, в которых при формировании новых решений (изменении порядков обработки данных в последовательностях $\pi_2^l$) гарантируется выполнение условия для левого дискретного градиента [2] $^-\nabla_v f(s)$ целевой функции $f$ в виде $^-\nabla_v f(s)\le 0$. Иначе говоря, $N_s^p$ – это множество направлений $v$, для которых гарантируется $^-\nabla_v f(s)\le 0$. Алгоритм определения эффективных порядков обработки данных $i\in N'_{пр}$ в последовательностях $\pi_2^l$ имеет следующую последовательность шагов:

1) начальное решение по составам и порядку обработки данных в последовательностях $\pi^l$ (последовательностях $\pi_2^l$, $l=\overline{1,L}$), полученное после размещения в $\pi_2^l$ данных *i*-х типов (для которых выполнялось условие $i\in N_{пр}\cap(\bigcup_{l=1}^{L}\pi_1^l)$), определим как локально эффективное и обозначим как $\pi^*(s)$, где $s$ – индекс текущего шага алгоритма (на начальной стадии его реализации $s$ = 0);

2) из множества $N'_{пр}$ выбирается тип данных $i'$ в соответствии с условием $i'=\min\{i\mid i\in N'_{пр}\}$ (на *s*-м текущем шаге алгоритма определяются эффективные местоположения данных $i'$ в последовательностях $\pi_2^l$, $i'\in\pi_2^l$);

3) для последовательностей $\pi_2^l$ ($l=\overline{1,L}$) выполняется проверка условия $i'\in\pi_2^l$, формируется множество $Ls$ номеров сегментов, для которых это условие выполняется (изменению подлежат позиции данных $i'$-го типа в тех последовательностях $\pi_2^l$, для которых $l\in Ls$,

возможные позиции рассматриваемых данных $i'$ в $\pi_2^l$: ($q$ + 1)-я, ($q$ + 2)-я,..., ($q$ + $n_l$)-я); в последовательностях $\pi_2^l$, для которых $l \in Ls$, определяются значения $j^l$ начальных позиций данных $i'$;

4) значение метрики $h$ окрестности $O_h(\pi(s))$ решения $\pi(s)$, в которой будет выполняться поиск локально эффективных решений, задается равным 1;

5) значение $v$ индекса шага промежуточного решения инициализируется 1 ($(s+v)$ – номер промежуточного шага алгоритма, связанного с определением локального эффективного решения по местоположению данных $i'$-го типа в последовательностях $\pi_2^l$, формируемого на основе локально эффективного расписания $\pi^*(s)$ (решения $(P^l(s)\ ; (t_{ji}^{0l}(s)))^*$);

6) для каждого сегмента $l$ (при $l \in Ls$) выполняется проверка условия $(q+1) < j^l$ (данные $i'$ занимают в $\pi^l(s)$ ($l \in Ls$) позицию $j^l$, одну из следующих за ($q$ + 1)-й позицией, и их положение в $\pi_2^l$ может быть изменено);

7) для $l$-х сегментов конвейера ($l \in Ls$), для последовательностей $\pi_2^l$ которых выполняется условие $(q+1) < j^l$, в зависимости от текущего значения окрестности $O_h(\pi(s))$ реализуется изменение положения данных $i'$-го типа в $\pi_2^l$ ($l \in Ls$) на одну позицию ближе к началу (в каждой последовательности $\pi_2^l$ ($l \in Ls$) при $h$ = *1*, в каждой паре последовательностей $\pi_2^l$ при $h$ = *2*, в трех последовательностях $\pi_2^l$ при $h$ = *3* и т. д.); так как порядок данных в последовательностях $\pi^l$ представляется матрицами $P^l$ ($l = \overline{1,L}$), тогда изменение $j^l$-й позиции рассматриваемых данных в последовательностях $\pi^l$ (в последовательностях $\pi_2^l$, $l \in Ls$) выполняется следующим образом: $p^l_{i',j^l-1}(s+v)=1$, $p^l_{i',j^l}(s+v)=0$, $p^l_{k',j^l-1}(s+v)=0$, $p^l_{k',j^l}(s+v)=1$, где $k'$ – индексы строк в матрицах $P^l$ ($l \in Ls$), в которых элементы $p^l_{k',j^l-1}$ в ($j^l-1$)-м столбце равны 1 (на $s$-м шаге алгоритма – это предыдущая позиция для рассматриваемых данных, которую они занимают на $(s+v)$-м шаге);

8) с использованием матриц $P^l$ ($l = \overline{1,L}$) вычисляются матрицы $(t_{ji}^{0l})$ ($l = \overline{1,L}$) для ($s$ + $v$)-го шага алгоритма, а также значения критерия $f(s+v)$ (в итоге формируются решения $(P^l(s+v), t_{ji}^{0l}(s+v) \mid l = \overline{1,L})$);

9) для исходного решения $((P^l(s)), (t_{ji}^{0l}(s)) \mid l = \overline{1,L})$ и каждого решения

$$((P^l(s+v)), (t_{ji}^{0l}(s+v)) \mid l = \overline{1,L})$$

вычисляются значения левых ${}^{-}\nabla_v f(s)$ либо правых ${}^{+}\nabla_v f(s)$ дискретных градиентов целевой функции $f$ [2]

$$({}^{-}\nabla_v f(s) = [f(s+v) - f(s)] \le 0,\ \text{б}) {}^{+}\nabla_v f(s) = [f(s+v) - f(s)] > 0;$$

в результате индексы $v$ групп последовательностей $\pi^l$, для которых выполняется ${}^{-}\nabla_v f(s) \le 0$, добавляются в множество $N_s^p$ : $N_s^p = N_s^p \cup \{v\}$, в итоге формируется множество $N_s^p$ тех направлений $v$, для которых ${}^{-}\nabla_v f(s) \le 0$; если $N_s^p = \varnothing$, то реализуется переход к шагу 14);

10) в множестве $N_s^p$ выбирается направление $v'$, для которого модуль отрицательного градиента ${}^{-}\nabla_v f(s) \le 0$ наибольший ($v' \to \max_v(|\ {}^{-}\nabla_v f(s) \le 0|)$; полученное решение

$$((P^l(s+v')), (t_{ji}^{0l}(s+v')) \mid l = \overline{1,L})$$

рассматривается как локально эффективное, на основе которого формируются последующие решения;

11) полученное решение $((P^l(s+v')),(t_{ji}^{0l}(s+v'))\,|\,l=\overline{1,L})$ фиксируется как локально эффективное: $((P^l(s)),(t_{ji}^{0l}(s))\,|\,l=\overline{1,L})^*$=$((P^l(s+v')),(t_{ji}^{0l}(s+v'))\,|\,l=\overline{1,L})$, фиксируется значение номера шага алгоритма $s=s+v$; индекс $j^l$ текущего столбца, идентифицирующий местоположение в $\pi_2^l$ рассматриваемых данных $i'$-го типа для последовательностей, измененных на текущем шаге, модифицируется ( $j^l=j^l-1$ );

12) выполняется проверка условия $(q+1)<j^l$ для каждой последовательности $\pi_2^l$ ($l\in Ls$);

13) при невыполнении условия $(q+1)<j^l$ для соответствующих последовательностей $\pi_2^l$ ($l\in Ls$), в которых был изменен порядок данных на ($s$ + $v$)-м шаге алгоритма, индексы этих последовательностей $l$ исключаются из множества $Ls$: $Ls=Ls\setminus\{l\}$, где номера $l$ соответствуют последовательностям, входящим в группу, идентифицируемую индексом *v'*, и для них выполняется условие $j'=(q+1)$ (где *v'* – индекс шага алгоритма, выполняемого относительно локально эффективного расписания $\pi^*(s)$); при выполнении условия $Ls\neq\varnothing$ реализуется переход к шагу 7; при выполнении условия $Ls=\varnothing$ отсутствуют последовательности $\pi^l$ (последовательности $\pi_2^l$, $l\in Ls$), в которых может быть изменен порядок обработки данных, выполняется переход к шагу 16;

14) если $N_s^p=\varnothing$, то в окрестности $O_h$ локально эффективного решения $((P^l(s)),(t_{ji}^{0l}(s))\,|\,l=\overline{1,L})^*$ (расписания $\pi^*(s)$) не найдено новое решение, уменьшающее значение целевой функции, тогда текущее значение метрики *h* модифицируется следующим образом: $h = h + 1$ – формируется модифицированное значение метрики окрестности $O_h(\pi^*(s))$, если $h\le h_{\max}$, то выполняется переход к шагу 6;

15) при выполнении условия $h>h_{\max}$ в окрестности $O_h$ текущего локально эффективного решения $((P^l(s)),(t_{ji}^{0l}(s))\,|\,l=\overline{1,L})^*$ более эффективное решение не найдено, поэтому локально эффективное решение $((P^l(s)),(t_{ji}^{0l}(s))\,|\,l=\overline{1,L})^*$ интерпретируется как эффективное решение по определению позиций данных $i'$-го типа в $\pi^l$ ($l=\overline{1,L}$); при реализации условия $h>h_{\max}$ выполняется переход к шагу 16;

16) выполняется модификация множества $N'_{пр}$ следующим образом. $N'_{пр}=N'_{пр}\setminus\{i'\}$, если для полученного в результате модификации множества $N'_{пр}$ выполняется условие $N'_{пр}=\varnothing$, тогда все данные *i*-х типов такие, что $i\in N'_{пр}$, размещены в последовательностях $\pi^l$ $(l=\overline{1,L})$ эффективным образом, тогда должен быть выполнен переход на шаг 17; если $N'_{пр}\neq\varnothing$, то не для всех типов данных выполнено определение эффективного их положения в $\pi^l$ $(l=\overline{1,L})$, реализуется переход на шаг 2;

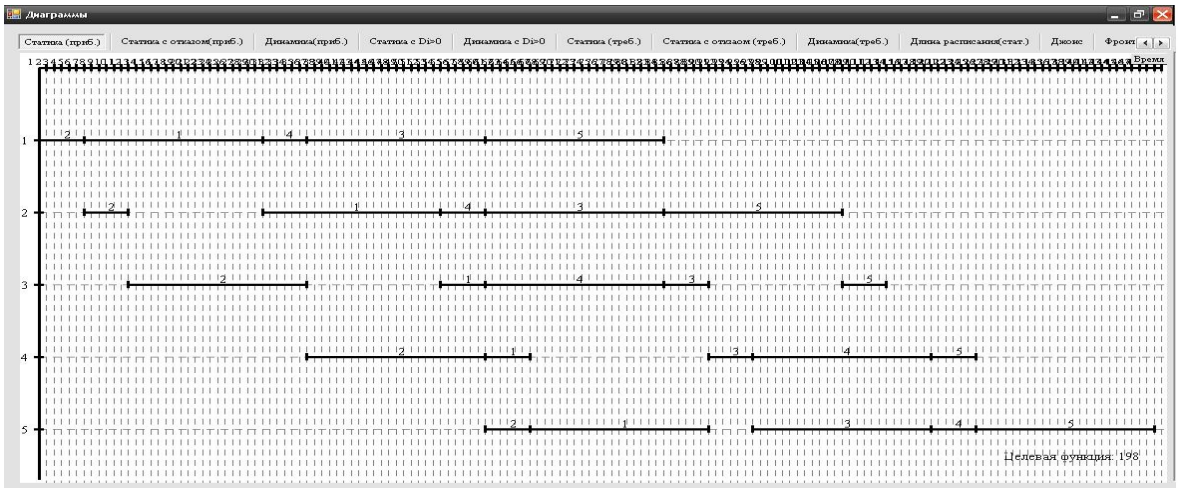
17) останов алгоритма.

В результате реализации сформулированного алгоритма в последовательностях $\pi^l$ $(l=\overline{1,L})$ (в последовательностях $\pi_2^l$) эффективным образом определены позиции данных *i*-х типов, для которых перед построением динамического расписания выполнялось условие $i\in N_{пр}\cap(\bigcup_{l=1}^{L}\pi_1^l)$. На втором этапе построения динамического расписания реализуется определение эффективного вида последовательностей $\pi_2^l$ ($l\in Ls$) при размещении в них данных *i*-х типов, для которых перед началом построения динамического расписания выполнялись условия $i\in N_{пр}\ \&\ (i\notin\bigcup_{l=1}^{L}\pi_1^l)$. Те же *i*-е типы данных входят в множество $N_{пр}$, полученное после реализации приведенного выше алгоритма размещения в последовательностях $\pi_2^l$ дан-

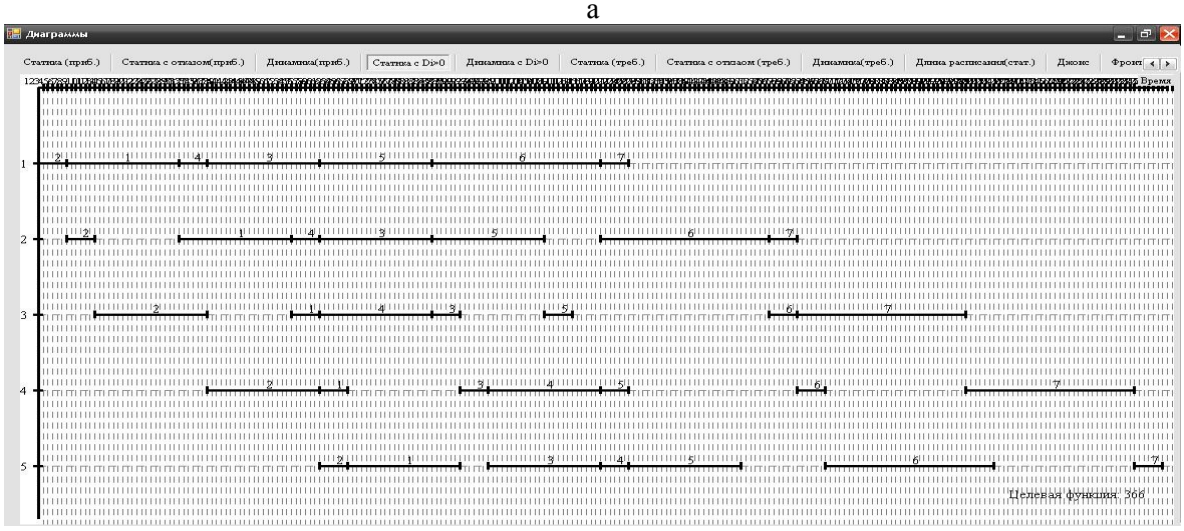
ных, для которых выполнялось условие $i \in N_{пр} \cap (\bigcup_{l=1}^{L} \pi_1^l)$. Так как данные, типы которых входят в полученное после реализации первого этапа алгоритма построения динамического расписания множество $N_{пр}$, должны быть добавлены в каждую из последовательностей $\pi_2^l$ ($l = \overline{1,L}$), тогда для определения их местоположения (позиции *j*) в этих последовательностях применен «жадный» алгоритм, предложенный в [1]. Отличием в использовании данного алгоритма при реализации второго этапа построения динамического расписания является то, что добавляемые на каждом шаге алгоритма в последовательности $\pi_2^l$ ($l = \overline{1,L}$) данные могут занять любую позицию в них, начиная с ($q$ + 1)-й (т. е. эффективное местоположение этих данных может быть определено только в пределах последовательностей $\pi_2^l$ ($l = \overline{1,L}$)). Рассмотренные первый и второй этапы построения динамических расписаний с учетом поступления на обработку в систему данных $i_{n+g} \in N_2$ в момент времени $d_{i_{n+g}} > 0$ позволяют определить модифицированные виды последовательностей $\pi^l$ ($l = \overline{1,L}$) при добавлении в них данных $i_{n+g}$. В ходе реализации первого и второго этапов метода построения динамического расписания для данных $i_{n+g}$ определено их эффективное местоположение в последовательностях $\pi^l$ ($l = \overline{1,L}$), тогда они удаляются из $N'_2$: $N'_2 = N'_2 \setminus \{i_{n+g}\}$. В случае выполнения условия $N'_2 \neq \varnothing$ из множества $N'_2$ извлекаются данные $i'_{n+g}$ в соответствии с условием $i'_{n+g} = \min_{d_{i_{n+g}}} \{i_{n+g} \mid i_{n+g} \in N'_2\}$, для которых определяется их эффективное местоположение в последовательностях $\pi^l$ ($l = \overline{1,L}$). Указанные действия по построению динамических расписаний для данных $i_{n+g}$ ($g = \overline{1,k}$) выполняются до тех пор, пока $N'_2 \neq \varnothing$. В случае $N'_2 = \varnothing$ все данные $i_{n+g}$ ($g = \overline{1,k}$), для которых выполняется условие $d_{i_{n+g}} > 0$, размещены эффективным образом в последовательностях $\pi^l$ ($l = \overline{1,L}$).

Предложенный метод построения динамических расписаний реализован программно. Результаты функционирования программы представлены на рис. 1 и 2. При этом каждый рисунок содержит вид исходного статического расписания, вид расписания после добавления в него данных с $d_{i_{n+g}} > 0$ (статическое расписание не перестраивается), вид динамического расписания, сформированного при добавлении в статическое расписание данных $i_{n+g}$ ($g = \overline{1,k}$). Для динамического расписания на рис. 1 заданы значения параметров: $n = 5$, $L = 5$, $k = 2$ ($g = \overline{1,2}$), а на рис. 2 – $n = 5$, $L = 5$, $k = 4$ ($g = \overline{1,4}$).

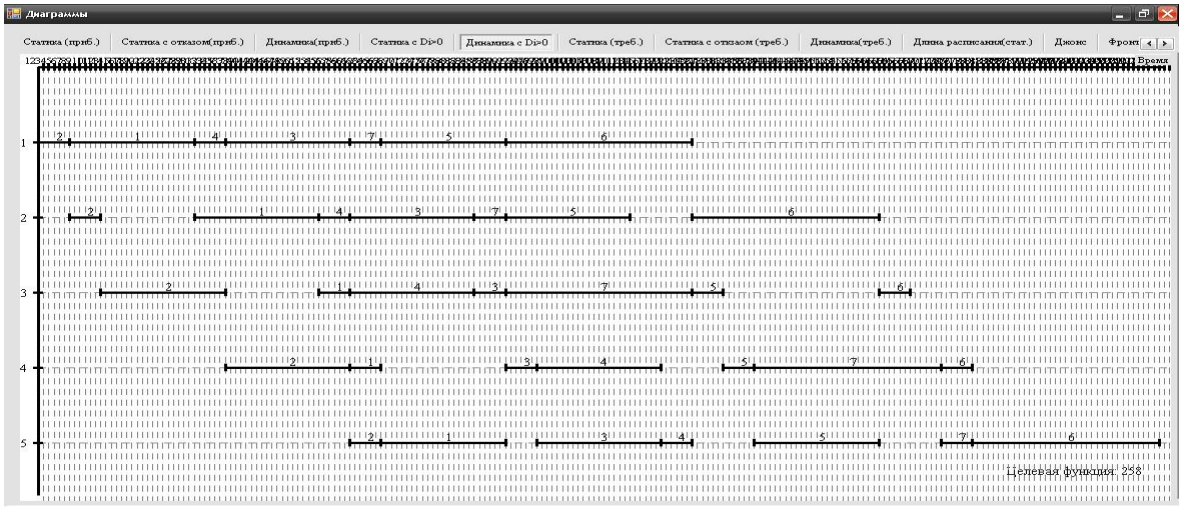
Для анализа эффективности сформулированного метода построения динамических расписаний в рассмотрение введены следующие обозначения параметров задачи: $\min(t_i^l)$ – минимальная длительность обработки данных *i*-х типов ($i = \overline{1,n}$) на *l*-х сегментах конвейера ($l = \overline{1,L}$), для которых $d_i = 0$ (данных в статическом расписании); $\max(t_i^l) / \min(t_i^l)$ – отношение максимальной длительности обработки данных *i*-х типов ($i = \overline{1,n}$) на *l*-х сегментах конвейера ($l = \overline{1,L}$), для которых $d_i = 0$, к минимальной длительности обработки данных этих типов, определяющее неоднородность длительностей обработки данных в статическом расписании; $\min(t_{i_{n+g}}^l)$ – минимальная длительность обработки данных ($i_{n+g}$)-х типов ($g = \overline{1,k}$) на *l*-х сегментах конвейера ($l = \overline{1,L}$), для которых $d_{i_{n+g}} > 0$ (данных, которые размещаются в сформированном статическом расписании при построении динамического расписания); $\max(t_{i_{n+g}}^l) / \min(t_{i_{n+g}}^l)$ – отношение максимальной длительности обработки данных *i*-х типов ($i = \overline{n+1, n+k}$) на *l*-х сегментах конвейера ($l = \overline{1,L}$), для которых $d_i > 0$, к минимальной дли-




а

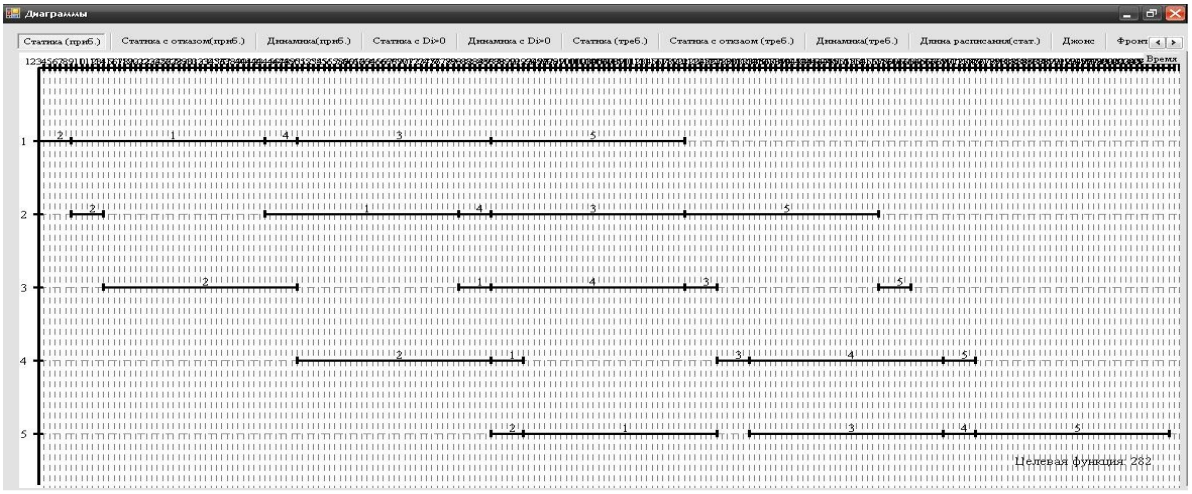



б

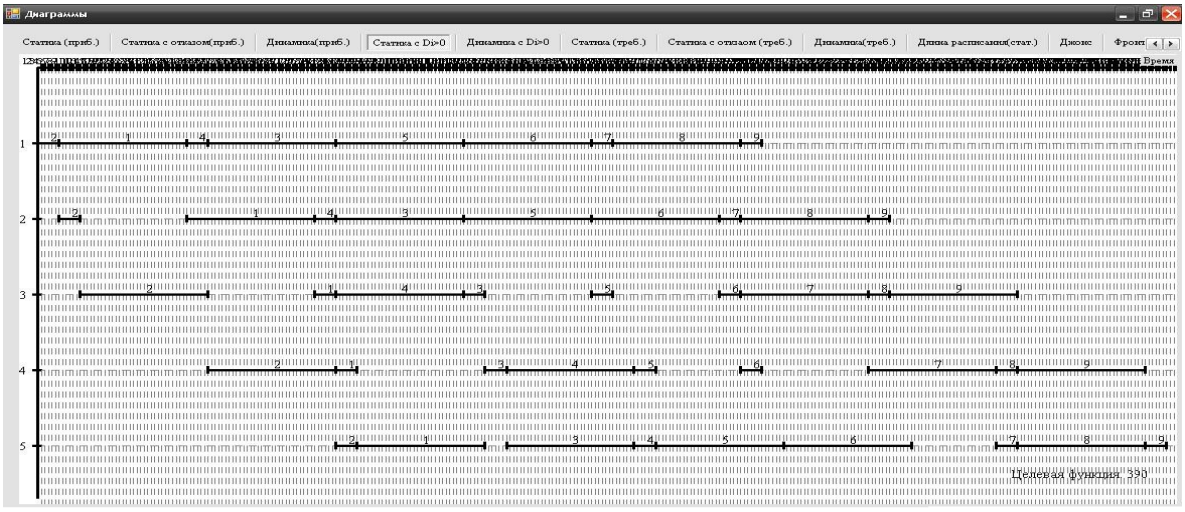



в

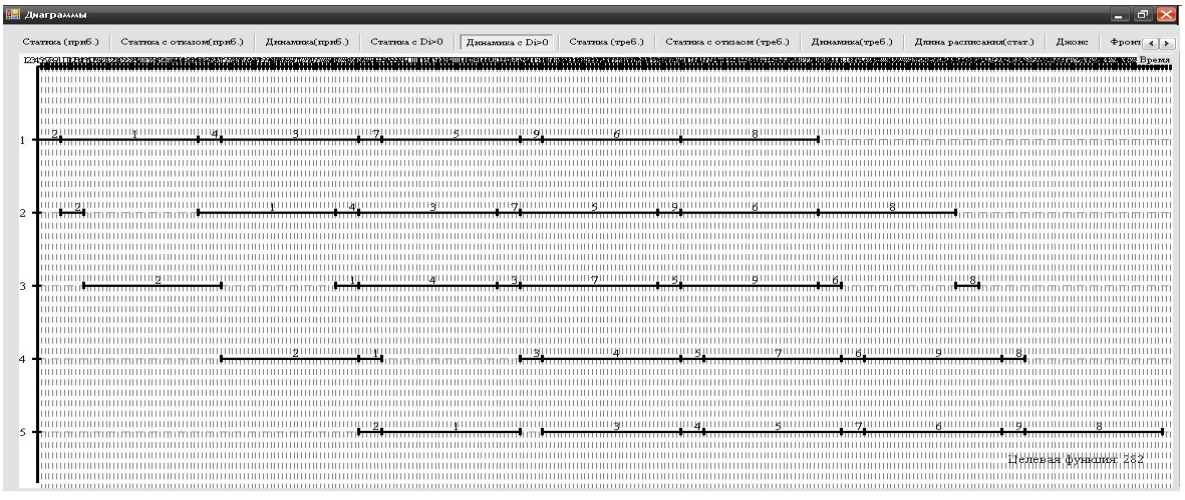
*Рис. 1.* Результаты функционирования программы построения динамических расписаний при $n = 5$; $L = 5$; $k = 2$ ($g = \overline{1,2}$): *а* – статическое расписание для данных с $d_i = 0$ ($i = \overline{1,n}$); *б* – статическое расписание, дополненное данными с $d_{i_{n+g}} > 0$ ($g = \overline{1,2}$); *в* – динамическое расписание, сформированное на основе статического, для данных с $d_{i_{n+g}} > 0$ ($g = \overline{1,2}$)




а




б



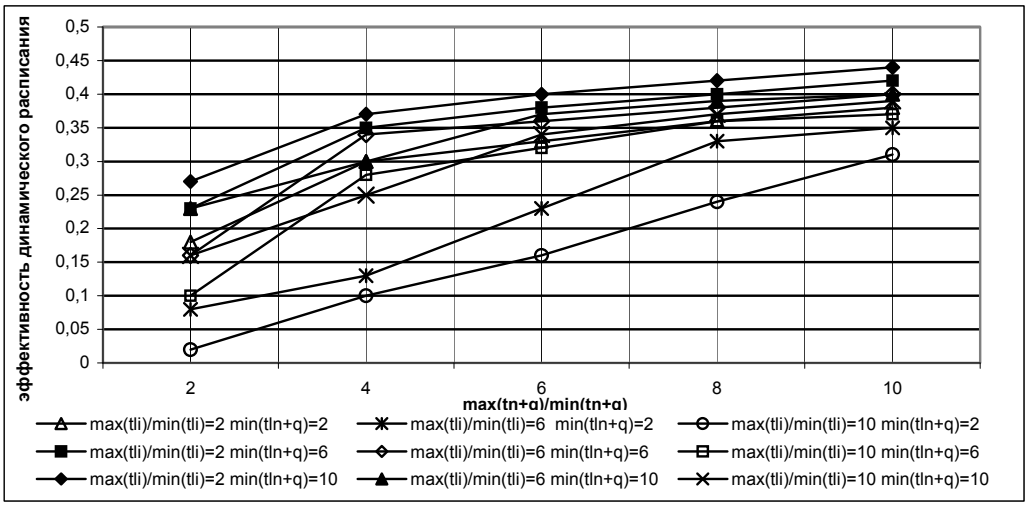

в

*Рис. 2.* Результаты функционирования программы построения динамических расписаний при $n=5$; $L=5$; $k=4$ ($g=\overline{1,4}$): *а* – статическое расписание для данных с $d_i=0$ ($i=\overline{1,n}$); *б* – статическое расписание, дополненное данными с $d_{i_{n+g}}>0$ ($g=\overline{1,4}$); *в* – динамическое расписание, сформированное на основе статического, для данных с $d_{i_{n+g}}>0$ ($g=\overline{1,4}$)

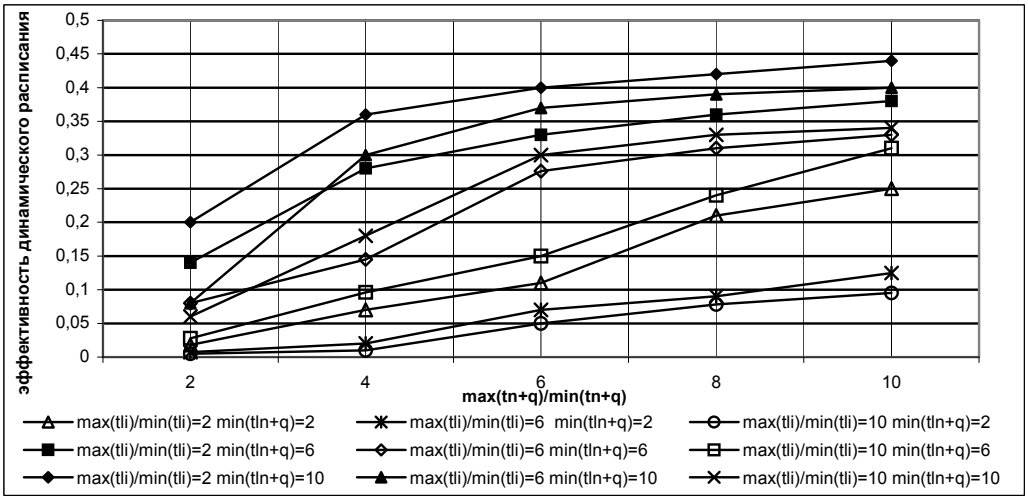
тельности обработки данных этих типов, определяющее неоднородность длительностей обработки данных в динамическом расписании. Исследование эффективности сформированных динамических расписаний выполнено при условии, что введенные параметры принимают следующие значения: а) $\min(t_i^l)$ – значения 2, 6, 10; б) $\max(t_i^l)/\min(t_i^l)$ – 2, 6, 10; в) $\min(t_{i_{n+g}}^l)$ – 2, 6, 10; г) $\max(t_{i_{n+g}}^l)/\min(t_{i_{n+g}}^l)$ – значения 2, 4, 6, 8, 10. Результаты исследований эффективности построения динамических расписаний для случаев $k=2$ и $k=4$ представлены на рис. 3, 4.

С целью анализа эффективности построения динамических расписаний результаты исследований были сформированы следующим образом. Для задаваемых значений параметров длительностей обработки данных *i*-х типов ($i=\overline{1,n}$) в статическом расписании и данных ($i_{n+g}$)-х типов ($g=\overline{n+1,n+k}$),поступающих на обработку в моменты времени $d_{i_{n+g}}>0$, программа формировала исходное статическое расписание для *n* типов данных, статическое расписание для *n* типов данных с добавленными в него данными с $d_{i_{n+g}}>0$, динамическое расписание для *n* типов данных с $d_i=0$ и *k* типов данных, для которых $d_{i_{n+g}}>0$. Для каждого вида расписаний вычислялись значения критерия *f*. Для определения эффективности построения динамических расписаний значения критерия, формируемые для статического расписания обработки *n* данных с добавленными к нему данными с $d_{i_{n+g}}>0$, обозначены как $f_c$, значения критерия, полученные для динамического расписания при определении для данных с $d_{i_{n+g}}>0$ их эффективного местоположения в последовательностях $\pi^l$ ($l=\overline{1,L}$), обозначены как $f_\partial$. Тогда эффективность построения динамического расписания определяется как $(f_c-f_\partial)/f_c$ (уменьшение простоев сегментов конвейера при формировании динамического расписания к суммарному значению простоев сегментов при статическом расписании). Анализ зависимостей эффективности формирования динамических расписаний от параметров длительностей обработки данных *i*-х ($i=\overline{1,n}$) и ($i_{n+g}$)-х типов ($g=\overline{n+1,n+k}$) позволил сделать следующие выводы. При значениях параметров $\min(t_i^l)=2$ и $\min(t_{i_{n+g}}^l)=10$ (при значениях отношения $\max(t_i^l)/\min(t_i^l)$ равных 2, 6, 10) эффективность предложенного метода построения динамических расписаний (эффективность сформированных динамических расписаний) максимальна.

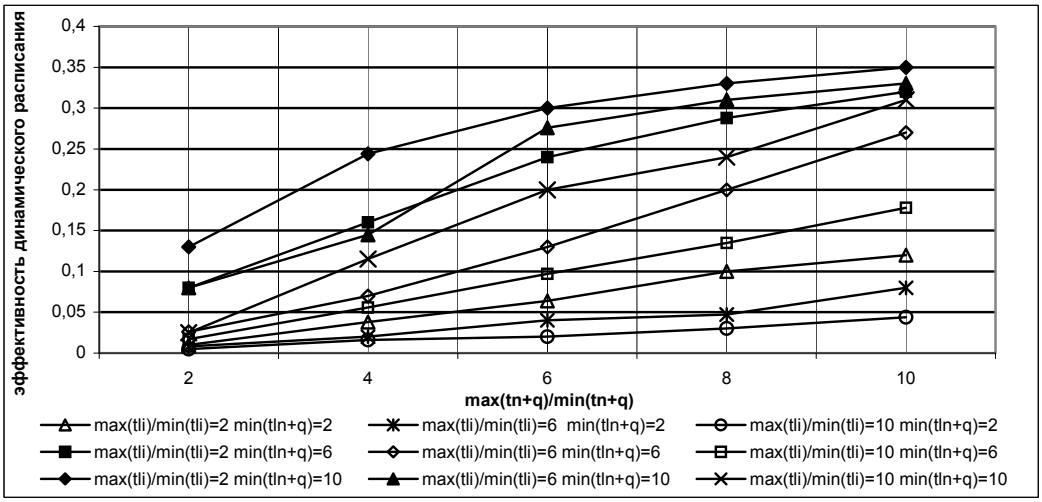
При этом максимум эффективности (для $\min(t_i^l)=2$, $\min(t_{i_{n+g}}^l)=10$) получен для значения $\max(t_i^l)/\min(t_i^l)=2$. Таким образом, формирование динамического расписания наиболее эффективно при малых длительностях обработки данных в статическом расписании (данные с $d_i=0$), малой неоднородности длительностей обработки этих данных ($\max(t_i^l)/\min(t_i^l)=2$), значительных длительностях обработки данных с $d_{i_{n+g}}>0$ ($\min(t_{i_{n+g}}^l)=10$). Изменение неоднородности длительностей обработки данных с $d_{i_{n+g}}>0$ от $\max(t_{i_{n+g}}^l)/\min(t_{i_{n+g}}^l)=2$ до $\max(t_{i_{n+g}}^l)/\min(t_{i_{n+g}}^l)=10$ обуславливает увеличение эффективности построения динамического расписания от 0,27 до 0,44. При значениях $\min(t_i^l)=2$, $\min(t_{i_{n+g}}^l)=10$ увеличение неоднородности длительностей обработки данных в статическом расписании (значения отношения $\max(t_i^l)/\min(t_i^l)$ равны 6 и 10) обуславливает уменьшение эффективности построения динамических расписаний. При $\max(t_i^l)/\min(t_i^l)=6$ эффективность построения динамического расписания находится в диапазоне от 0,23 до 0,4; при $\max(t_i^l)/\min(t_i^l)=10$ – в диапазоне от 0,16 до 0,38. Таким образом, при увеличении неоднородности длительностей обработки данных в статическом расписании эффективность построения динамического расписания снижается на значения, находящиеся в диапазоне от 0,04 до 0,11. Уменьшение длительностей




а

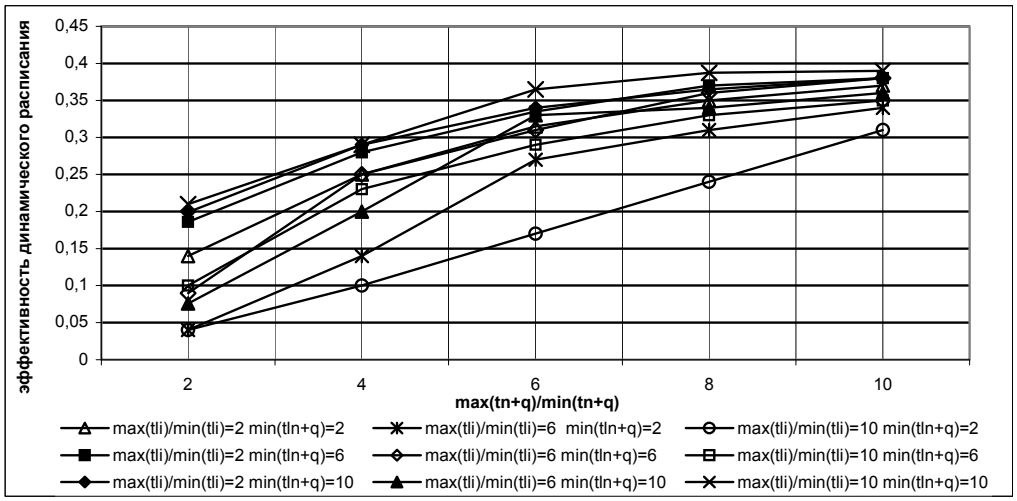



б

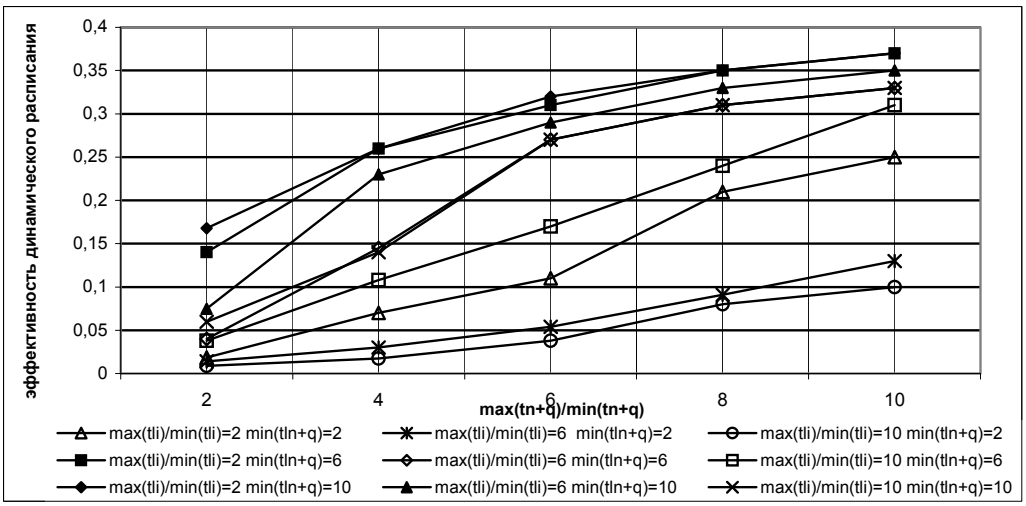



в

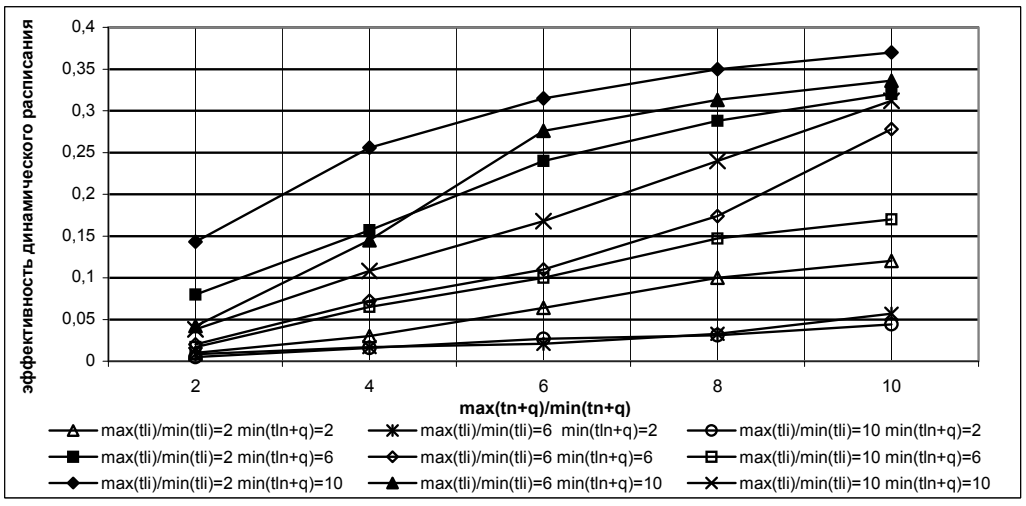
*Рис. 3*. Эффективность метода построения динамических расписаний обработки данных при различных моментах времени их поступления и $q = \overline{1,2}$: *а* – эффективность динамического расписания при $\min(t_i^l) = 2$ ($i = \overline{1,n}$); *б* – эффективность динамического расписания при $\min(t_i^l) = 6$ ($i = \overline{1,n}$); *в* – эффективность динамического расписания при $\min(t_i^l) = 10$ ($i = \overline{1,n}$)




а




б




в

*Рис. 4*. Эффективность метода построения динамических расписаний обработки данных при различных моментах времени их поступления и $q = \overline{1,4}$: *а* – эффективность динамического расписания при $\min(t_i^l) = 2$ ($i = \overline{1,n}$); *б* – эффективность динамического расписания при $\min(t_i^l) = 6$ ($i = \overline{1,n}$); *в* – эффективность динамического расписания при $\min(t_i^l) = 10$ ($i = \overline{1,n}$)

обработки данных с $d_{i_{n+g}} > 0$ (значения $\min(t^l_{i_{n+g}}) = 6$ и $\min(t^l_{i_{n+g}}) = 2$) обуславливает уменьшение эффективности построения динамических расписаний по сравнению с расписаниями, сформированными при $\min(t^l_{i_{n+g}}) = 10$ и теми же значениями остальных параметров. При $\min(t^l_{i_{n+g}}) = 6$ эффективность сформированных динамических расписаний снижается на 0,05 (для $\max(t^l_i) / \min(t^l_i) = 2$) и 0,16 (для $\max(t^l_i) / \min(t^l_i) = 10$). При $\min(t^l_{i_{n+g}}) = 2$ эффективность сформированных динамических расписаний снижается на 0,1 (для $\max(t^l_i) / \min(t^l_i) = 2$) и 0,25 (для $\max(t^l_i) / \min(t^l_i) = 10$).

Таким образом, максимальная эффективность построения динамических расписаний наблюдается при малых значениях длительностей обработки данных в статическом расписании (при $d_i = 0$) и больших значениях длительностей обработки данных, поступающих в систему в моменты времени $d_{i_{n+g}} > 0$ (при условии значительной неоднородности длительностей обработки данных ($i_{n+g}$)-ых типов ($g = \overline{n+1, n+k}$)). При уменьшении длительностей обработки данных, поступающих на обработку в моменты времени $d_{i_{n+g}} > 0$, эффективность сформированных динамических расписаний снижается.

Увеличение длительностей обработки данных *i*-х типов ($i = \overline{1, n}$) (соответственно значения $\min(t^l_i) = 6$ и $\min(t^l_i) = 10$) при тех же (рассматриваемых для случая $\min(t^l_i) = 2$) значениях остальных параметров обуславливает уменьшение эффективности сформированных динамических расписаний. Эффективность динамических расписаний для случая $\min(t^l_i) = 6$ снижается (по сравнению с экспериментами с $\min(t^l_i) = 2$) на значения, находящиеся в диапазоне от 0,002 до 0,07, для случая $\min(t^l_i) = 10$ – на значения, находящиеся в диапазоне от 0,01 до 0,12. При этом снижение эффективности динамических расписаний более значительно при малых значениях отношения $\max(t^l_{i_{n+g}}) / \min(t^l_{i_{n+g}})$, при увеличении значений этого отношения до 8, 10 снижение эффективности незначительно.

На рис. 4 представлены виды графических зависимостей эффективности динамических расписаний от входных параметров задачи для количества *k* добавляемых в статические расписания данных, равного 4 (*k* = *4*). Большее количество данных $i_{n+g}$ при тех же значениях параметра неоднородности длительностей их обработки приводит к незначительному снижению эффективности построения динамических расписаний по сравнению со случаем (*k* = 4). Максимальное снижение эффективности динамических расписаний наблюдается при малых значениях параметра $\max(t^l_{i_{n+g}}) / \min(t^l_{i_{n+g}})$, увеличение значений этого параметра до 8, 10 обуславливает меньшее снижение эффективности метода построения динамических расписаний. Незначительное снижение эффективности построения динамических расписаний связано с увеличением количества обрабатываемых в системе данных (с $d_{i_{n+g}} > 0$) при неоднородности длительностей их обработки.

**Заключение**

Разработан метод построения динамических расписаний обработки данных в конвейерной системе при различных моментах времени поступления их на обработку, позволяющий определять локально эффективные решения. Предложены условия, которые позволяют идентифицировать возможные позиции данных, поступающих в систему в моменты времени $d_{i_{n+g}} > 0$, в последовательностях обработки на каждом из приборов. Выполненные исследования показали эффективность формирования динамических расписаний по сравнению со статическими расписаниями, предполагающими добавление поступающих в моменты времени $d_{i_{n+g}} > 0$ данных в конец каждой из последовательностей обработки данных на сегментах

конвейера. Простои сегментов конвейера в динамическом расписании до 45 % меньше, чем простои сегментов при реализации статического расписания с добавляемыми в конец последовательностей данными. Возможными направлениями дальнейших исследований является адаптация предложенного градиентного метода к решению задач теории расписаний, учитывающих влияние возмущающих воздействий на запланированный в соответствии со статическим расписанием ход вычислительного процесса.

**Список литературы**

1. *Кротов К. В.* Градиентный метод составления статических расписаний для конвейерных систем, основывающийся на жадных стратегиях // Вестн. Новосиб. гос. ун-та. Серия: Информационные технологии. 2015. Т. 13, вып. 1. С. 55–73.

2. *Ковалев М. М.* Матроиды в дискретной оптимизации. М.: Едиториал УРСС, 2003. 224 с.

3. *Кочетов Ю. А.*, *Младенович Н.*, *Хансен П.* Локальный поиск с чередующимися окрестностями // Дискретный анализ и исследование операций. 2003. Т. 10, № 1. С. 11–43.

4. *Кононова П. А.* Нижние и верхние оценки длины оптимального расписания презентаций медиаобъектов // Дискретный анализ и исследование операций. 2012. Т. 19, № 1. С. 54–73.

5. *Кононова П. А.* Локальный поиск с чередующимися окрестностями для задачи Джонсона с пассивным буфером // Дискретный анализ и исследование операций. 2012. Т. 19, № 5. С. 63–82.

6. *Кононова П. А.* Локальный поиск с чередующимися окрестностями для задачи Джонсона с пассивным буфером: Дис. … канд. физ.-мат. наук. Новосибирск, 2012. 106 с.

*Материал поступил в редколлегию 15.09.2015*

**K. V. Krotov**

*Sevasopol State University*
*33 Universitetskaya Str.*, *Sevastopol*, *299053*, *Russian Federation*

*krotov_k1@mail.ru*

## GRADIENT METHOD OF CREATING THE DYNAMIC SCHEDULING OF PROCESSING DATA IN A CONVEYOR SYSTEM AT DIFFERENT POINTS IN TIME OF THEIR RECEIPT

Model of construction scheduling of processing data in conveyor system and method of constructing dynamic scheduling data at different time points of entry into the system are grounded in work.

*Keywords*: conveyor system, schedule program execution, dynamic scheduling, data acquisition times for processing locally effective solution neighborhood .

**References**

1. Krotov K. V. Gradient method of drawing up schedules for static conveyor systems, based on greedy strategy. *Vestnik Novosibirsk State University. Series: Information technology*, 2015, vol. 13, № 1, p. 55–73.

2. Kovalev M. M. Matroids in discrete optimization. Moscow, Editorial URSS Publ., 2003, 224 p.

3. Kochetov Y. A. Local search with alternating neighborhoods / Y. A. Kochetov, N. Mladenovic, P. Hansen. *Discrete Analysis and Operations Research*, 2003, vol. 10, № 1, p. 11–43.

4. Kononova P. A. Lower and upper bounds for the length of the optimal schedule of presentations of media objects. *Discrete Analysis and Operations Research*, 2012, vol. 19, № 1, p. 54–73.

5. Kononova P. A. Local search with alternating neighborhoods for the Johnson problem with passive buffer. *Discrete Analysis and Operations Research*, 2012, vol. 19, № 5, p. 63–82.

6. Kononova P. A. Local search with alternating neighborhoods for the Johnson problem with passive buffer. Sci. Diss. Novosibirsk, 2012, 106 p.