УДК 004: 519.854

**К. В. Кротов**

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

*krotov_k1@mail.ru*

# ГРАДИЕНТНЫЙ МЕТОД СОСТАВЛЕНИЯ СТАТИЧЕСКИХ РАСПИСАНИЙ ДЛЯ КОНВЕЙЕРНЫХ СИСТЕМ, ОСНОВЫВАЮЩИЙСЯ НА ЖАДНЫХ СТРАТЕГИЯХ

Обосновываются модель составления расписаний обработки данных в конвейерных системах и метод построения статических расписаний, основывающийся на жадных стратегиях.

*Ключевые слова*: многостадийная конвейерная система, расписания выполнения программ обработки данных, градиентный метод, жадный алгоритм.

## Введение

Современные способы повышения производительности выполнения программ при обработке данных связаны с использованием вычислительных кластеров и распределенных вычислительных систем – GRID-систем. Повышение производительности выполнения программ связано с их параллельной реализацией в составе кластера либо с распределенным выполнением в GRID-системах. Одним из возможных способов повышения производительности выполнения программ обработки данных в составе кластера является их (программ) конвейеризация т. е. обработка данных соответствующими им программами осуществляется в составе программного макроконвейера. Конвейеризированное выполнение программ (обработка данных) соответствует классу программных систем МПОД [1] (много программ – одни данные), что предполагает организацию обработки одного потока данных последовательностью программ (последовательностью фрагментов программ) – реализацию программного конвейера. Развитие идей конвейеризации выполнения программ предполагает обработку потоков данных различных типов (в общем случае $n$ типов) соответствующими каждому из этих типов данных программами при реализации последовательного обмена данными между вычислительными сегментами программного конвейера (обмен данными между программами выполняется только после окончания их обработки на сегментах конвейера).

Конвейеризация программ предполагает их разделение на фрагменты, каждый из которых закреплен для выполнения за соответствующим сегментом конвейера (вычислительным устройством, обрабатывающим прибором). Введем в рассмотрение следующие обозначения: $i$ – номер множества однотипных данных, характеризующих одинаковые объекты, которые должны быть обработаны в системе; $I$ – множество всех данных, которые будут обработаны в вычислительной системе; $n$ – количество множеств данных, тогда $i = \overline{1,n}$); $n^i$ – количество элементов в множестве однотипных данных, характеризуемых индексом $i$ (множество со-

*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.

держит данные об $n^i$ одинаковых объектах). Данные, входящие в некоторое *i*-е множество, обрабатываются соответствующей им программой. Индекс *i* соответствует программе, выполняемой в составе конвейера, обрабатывающей данные *i*-го типа (соответствует типу выполняемой в составе конвейера программы, 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. Цель функционирования конвейерной системы в этом случае состоит в обработке поступающих на ее вход данных, выполняющейся в системе конвейеризированными программами. При этом предполагается, что выполняющие обработку данных конвейеризированные программы находятся в оперативной памяти каждого из сегментов конвейера (загружены в оперативную память вычислительных устройств кластера). 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. Определение подхода по конвейеризации обработки данных выполняющимися в многостадийных системах программами рассматривается в [2–4].

Постановка задачи управления вычислительным процессом, рассматриваемой в данной работе, предполагает, что при $n^i = 1$ ($i = \overline{1,n}$) реализуется обработка единичных данных (однократный запуск на выполнение соответствующих программ), для действий с которыми должно быть сформировано расписание выполнения программ.

**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 определения эффективных решений. В работах [5; 6] [1] и [9] выполняется решение классических задач теории расписаний обработки единичных данных для одного либо нескольких сегментов конвейера при различных видах критериев оптимизации и наличии ограничений на директивные сроки окончания обслуживания. При этом развиваются как точные (ветвей и границ, ветвей и отсечений), так и приближенные методы получения расписаний выполнения программ обработки данных. В работе [2] выполняется решение задач теории расписаний без наличия ограничений на порядок выполнения операций с данными, предполагающих произвольный маршрут обработки. При этом развиваются точные методы построения расписаний обработки единичных данных для ограниченного числа приборов (2–4 прибора) – метод ветвей и границ, а также приближенные методы – применение генетических алгоритмов к решению задач теории расписаний. В конечном итоге работа В. Г. Кобака посвящена решению статических задач размещения обрабатываемых единичных данных на обрабатывающих приборах в многостадийной системе. Применению эвристических и приближенных методов посвящены также исследования [7] [3], в которых рассматриваются задачи формирования статических расписаний обработки единичных данных. Развитию классических постановок теории расписаний посвящены работы иностранных авторов

[1] *Лазарев А. А.* Методы и алгоритмы решения задач теории расписаний для одного и нескольких приборов и их применение для задач комбинаторной оптимизации. Дисс. ... д-ра физ.-мат. наук. М., 2007. 426 с.; *Садыков Р. Р.* Алгоритмы решения задач теории расписаний для одного прибора с критериями $L_{\max}$ и $\sum w_j U_j$ : Дисс. … канд. физ.-мат. наук. М., 2006. 131 с.

[2] *Кобак В. Г.* Методология составления аналитической оценки распределительных задач и средства ее программно-алгоритмической поддержки: Дисс. … д-ра техн. наук. Ростов н/Д, 2008. 317 с.

[3] *Гончар Д. Р.* Методы планирования вычислений в САПР систем реального времени. Дисс. … канд. техн. наук. М., 2008. 139 с.

(в частности, [8]). Решение задач групповой обработки (обработки данных в партиях) рассматривается в работе [9], однако описанные там методы построения расписаний 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, так же как не выполняется при построении расписаний учет ресурсов самой системы, используемых ею при обработке данных (о возможности реализации построения расписаний при многих критериях и о возможной иерархии критериев упоминается в работе [6]).

**Постановка цели и задач научного исследования**

Цель работы состоит в совершенствовании методов построения расписаний обработки единичных данных в конвейерных системах. Совершенствование методов построения расписаний обработки единичных данных связано с учетом влияния возмущающих воздействий на ход вычислительного процесса, что предполагает формирование динамических расписаний, в которых порядок выполнения программ должен быть изменен по сравнению со статическим расписанием в зависимости от реализованного события в системе. Таким образом, основой для формирования динамических расписаний, учитывающих происходящие в системе события, являются статические расписания выполнения программ обработки данных. Тогда для решения задачи разработки методов построения динамических расписаний должна быть решена задача обоснования метода формирования статических расписаний, модификация которого позволит формировать и динамические расписания. Для достижения поставленной цели в статье решается задача, связанная с обоснованием градиентного метода построения статических расписаний выполнения программ обработки данных, реализующего «жадные» стратегии в дискретной оптимизации. При этом доказывается возможность применения жадного подхода к построению статических расписаний обработки единичных данных.

**Основное содержание работы**

Рассматриваемая задача является задачей с полной информацией, т. е. все параметры, характеризующие обрабатываемые данные (типы данных, количество данных каждого типа, длительности обработки данных различных типов на приборах и т. д.) и функционирующую систему (количество обрабатывающих приборов, дисциплину обработки данных и т. д.) являются задаваемыми. Если через $i$ обозначен идентификатор типа данных, обрабатываемых в системе, и, соответственно, идентификатор типа программы, выполняющей обработку этих данных, тогда через $d_i$ обозначен момент времени поступления в систему каждого $i$-го типа данных ($i = \overline{1,n}$). Для всех программ моменты времени их загрузки в систему и моменты поступления данных на обработку одинаковы, при этом $d_i = 0$. Обозначим через $l$ индекс обрабатывающего прибора, входящего в состав многостадийной системы ($l$-й сегмент вычислительной конвейерной системы), осуществляющего выполнение $l$-й части программы в системе, 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 выполняемых в системе программ предполагает прохождение данными, которые они обрабатывают, всех сегментов конвейера, при этом если $l$-й прибор приступил к выполнению $i$-й программы (к обработке данных $i$-го типа), обработка не может быть прервана. Все обрабатывающие приборы конвейерной системы характеризуются равными и неизменными во времени значениями производительности их работы. Выполнение на каждом $l$-м приборе назначенной ему части $i$-го программы характеризуется параметром длительности обработки данных на этом приборе, однозначно соответствующей объему выполняемых вычислений при интерпретации программного кода. В соответствии с выполненными рассуждениями в рассмотрение введено множество $\boldsymbol{N}$ типов данных ($\boldsymbol{N} = \{1,2,..,n\}$, $n$ – количество типов данных), обработка которых реализуется в сис-

теме; так как в системе выполняется обработка единичных данных, то $n^i = 1$ (при $i = \overline{1,n}$), т. е. для каждой из $n$ обрабатывающих программ будет выполнен однократный ее запуск на выполнение.

Расписание обработки единичных данных (запуска на выполнение программ) обозначим как $\pi$. Расписание обработки данных (выполнения программ) $\pi$ – это совокупность последовательностей $\pi^l$ запуска данных на обработку на каждом $l$-м сегменте конвейера ($l = \overline{1, L}$), оно имеет вид $\pi = \{\pi^1, \pi,^2 \pi^3, ..., \pi^L\}$. Так как порядки запуска на выполнение программ (последовательности $\pi^l$ ($l = \overline{1, L}$)) различны, то 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, размеры матриц $n \times n$, где $n$ – количество типов данных.

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. Так как перед началом обработки данных все выполняющиеся программы уже загружены в оперативную память сегментов конвейера и непосредственно при поступлении данных в момент времени $d_i = 0$ начинается их обработка, то длительности первоначальной наладки сегментов конвейера на обработку данных $i$-х типов не учитываются. 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 ($i = \overline{1,n}$) на $l$-х приборах системы ($l = \overline{1, L}$). Значения элементов матриц $(t_{ji}^{01})$ ($l = \overline{1, L}$) определяются в соответствии с видом матриц $(\boldsymbol{P})^l$ ($l = \overline{1, L}$) 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 $(\boldsymbol{P})^1$ следующим образом:

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

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

$$t_{ki}^{01} = \sum_{j=1}^{k-1}\sum_{h=1}^{n} t_h^1 \cdot p_{h,j}^1 \text{ , при } k > 1, \qquad (1)$$

где $k$ – номер позиции данных $i$-го типа в последовательности $\pi^1$.

Для $l$-го прибора (при $l \neq 1$) элементы матрицы $(t_{ji}^{0l})$ определяются выражениями вида:

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

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

где индекс $i$ типа данных определяется по матрице $(\boldsymbol{P})^l$ ($l = \overline{1, L}$) 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 ($j \neq l$) в матрице $(\boldsymbol{P})^l$ ($l = \overline{2, L}$) для данных $i$-го типа:

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

Сформированные 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 различаются в зависимости от видов сформированных решений, которые представляются в форме $[(\boldsymbol{P})^l, (t_{ji}^{0l}) \mid l = \overline{1, L}]$. Для анализа эффективности получаемых решений должен быть сформирован критерий, учитывающий рассчитываемые с использованием выражений (1)–(3) значения характеристик вычислительного процесса.

Особенности алгоритма метода определения порядка обработки данных, используемого при построении решений (расписания выполнения программ) и реализующего «жадную» стратегию:

1) добавление в конец сформированных на предыдущем ((s–1)-м) шаге алгоритма последовательностей выполнения программ $\pi^l(s-1)$ 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 $\pi^l$ текущих рассматриваемых данных 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 их 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 ($n - i$) 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 од-

ного 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. Оценка эффективности формируемых решений возможна с точки зрения внутренней и внешней целей функционирования системы. Внешняя цель функционирования системы определяет необходимость обработки данных таким образом, чтобы общее время вычислительного процесса было минимальным. Минимизация простоев программ обработки данных в ожидании готовности сегментов конвейера гарантирует выполнение внешней цели системы (т. е. выполнение внешней цели функционирования системы оценивается на основе анализа простоев программ обработки данных). Внутренняя цель функционирования системы определяет необходимость эффективного использования ее оборудования, т. е. минимизацию его (оборудования) простоев. В качестве цели, которая учитывается при формировании вида критерия, рассматривается внутренняя цель функционирования системы. Для обоснования вида критерия эффективности формируемого решения выполнены следующие рассуждения:

1) при $\sum_{i=1}^{n} t_{ji}^{0l} \cdot p_{ij}^{l} > \sum_{i=1}^{n} (t_{j-1,i}^{0l} + t_i^l) \cdot p_{i,j-1}^{l}$ *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); тогда длительность простоев *l*-го сегмента конвейера при выполнении текущего количества программ обработки данных в последовательности $\pi^l(s)$ определяется выражением $\sum_{j=2}^{n}\sum_{i=1}^{n} t_{ji}^{0l} \cdot p_{ij}^{l} - (t_{j-1,i}^{0l} + t_i^l) \cdot p_{i,j-1}^{l}$; полученное выражение позволяет определять суммарный простой *l*-го сегмента в ожидании готовности данных при их обработке;

2) для последовательности $\pi^l$ простои первого сегмента конвейера в ожидании готовности данных отсутствуют, тогда суммарное время простоев всех $L$ сегментов конвейерной системы, связанных с ожиданием готовности данных для обработки, будет определено выражением вида $\sum_{l=2}^{L}\sum_{j=2}^{n}\sum_{i=1}^{n} t_{ji}^{0l} \cdot p_{ij}^{l} - (t_{j-1,i}^{0l} + t_i^l) \cdot p_{i,j-1}^{l}$;

3) простои сегментов конвейера при выполнении программ связаны также с ожиданием поступления данных на обработку, если эти данные занимают первую позицию в последовательностях $\pi^l$ ($l = \overline{2, L}$), для первого сегмента конвейера ожидание начала обработки данных отсутствует, так как $t_{1i}^{01} = 0$; 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 $\sum_{i=1}^{n} t_{1,i}^{0l} \cdot p_{i1}^{l}$; тогда простои всех *l*-х сегментов конвейера (при $l = \overline{2, L}$) будут определены в соответствии с выражением $\sum_{l=2}^{L}\sum_{i=1}^{n} t_{1,i}^{0l} \cdot p_{i1}^{l}$.

Исходя из описанных рассуждений общий (суммарный) простой всех сегментов конвейера при обработке текущего количества данных определяется суммарным их простоем в ожидании готовности данных в первой позиции в $\pi^l$ ($l = \overline{2, L}$), а также суммарным их просто-

ем в ожидании готовности данных в некоторой *j*-й позиции (при условии, что $j \neq l$). Тогда конечный вид выражения для критерия эффективности формируемых решений по порядку выполнения программ (порядку обработки данных) в последовательностях $\pi^l$ ($l = \overline{2, L}$) следующий:

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

Обозначим через *O* метрику окрестности текущего рассматриваемого решения $\pi(s)$, в которой выполняется поиск нового локально эффективного решения, а через $O_{max}$ – максимально возможную метрику окрестности текущего рассматриваемого решения. Значение метрики *O* окрестности решения определяется выражением вида

$$O(s) = \sum_{i=1}^{n}\sum_{j=1}^{n} | p_{i,j}(s+g) - p_{i,j}(s) | / 2,$$

где *g* – индекс промежуточного шага алгоритма, на котором выполняется формирование нового решения $\pi(s+g)$ на базе локально эффективного (исходного) решения $\pi(s)$. Начальное значение метрики окрестности $O(s)$ задается равным 1. При $O(s) = l$ положение данных рассматриваемого *i*-го типа изменяется в одной из последовательностей $\pi^l$ ($l = \overline{1, L}$), при $O(s) = 2$ положение данных *i*-го типа изменяется в двух последовательностях $\pi^l$ ($l = \overline{1, L}$) и т. д.

Назовем алгоритм построения расписаний обработки единичных данных с использованием жадных стратегий Greedy Sheduler. Алгоритм определения эффективного положения обрабатываемых данных *i*-го типа в последовательностях $\pi^l$ ($l = \overline{1, L}$) Greedy Sheduler 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 окрестности $O(s)$ (в одной последовательности $\pi^l$ ($l = \overline{1, L}$), в двух последовательностях $\pi^l$, в трех последовательностях и т. д. в зависимости от значения метрики окрестности $O(s)$ текущего рассматриваемого решения);

3) изменение положения данных *i*-го типа в последовательностях $\pi^l$ обуславливает изменение значений целевой функции *f*, при этом направление дискретного градиента функции *f*, обозначенное как *g*, сопоставляется с идентификаторами последовательностей $\pi^l$, в которых изменяется положение рассматриваемых данных ($l = \overline{1, L}$, рассматриваются левый $^{-}\nabla f_g \leq 0$ либо правый $^{+}\nabla f_g > 0$ дискретные градиенты целевой функции *f* [14]), т. е. $^{-}\nabla f_g(\pi(s)) \leq 0$ – левый дискретный градиент целевой функции $f$, соответствующий расписанию $\pi(s+g)$, 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 $^{-}\nabla f_g \leq 0$ вдоль направлений *g*, то среди всех *g*, для которых выполняется это условие, выбирается то направление *g'*, для которого $\max_g(\left|^{-}\nabla_g f(\pi(s)) \leq 0\right|)$;

5) в тех последовательностях $\pi^l$, в которых изменение положений данных *i*-го типа гарантирует выполнение условия $\max_g(\left|^{-}\nabla_g f(\pi(s)) \leq 0\right|)$, позиции данных фиксируются (т. е.

рассматриваемое 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 с меньшей метрикой $O(s)$ 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 окрестности для формируемых решений не превысит максимально возможное значение $O_{max}$, увеличение метрики окрестности выполняется до тех пор, пока $O(s) \le O_{\max}$).

Таким образом, направление *g* дискретного градиента целевой функции *f* соответствует идентификаторам последовательностей $\pi^l$, в которых изменяется положение данных рассматриваемого *i*-го типа на (*s* + *g*)-м шаге алгоритма, т. е. направление *g* соответствует индексу шага *g* определения эффективного решения на основе текущего локально эффективного решения $\pi(s)$ и соответствует некоторой *g*-й группе последовательностей $\pi^l$, в которых на данном шаге алгоритма одновременно изменяется положение данных рассматриваемого *i*-го типа.

Для обоснования алгоритма формирования статических расписаний обработки данных Greedy Sheduler в рассмотрение введены следующие обозначения: s – номер текущего шага алгоритма; *i′* – индекс типа данных, местоположение которых в последовательностях $\pi^l$ ($l = \overline{1,L}$) определяется на текущем шаге алгоритма; $v^{\max}$ – индекс столбцов матриц $(P)^l$ ($l = \overline{1,L}$), в *i′*-ые строки которых добавляются значения 1, соответствующие текущему рассматриваемому *i′*-му типу данных (индекс столбца $v^{max}$ соответствует концам последовательностей $\pi^l$ ($l = \overline{1,L}$), где размещаются данные *i′*-го типа, эффективное местоположение которых в $\pi^l$ определяется); *v* – индекс столбца матрицы $(P)^l$ ($l = \overline{1,L}$), соответствующего текущему местоположению в $\pi^l$ данных рассматриваемого *i′*-го типа (индекс текущей позиции данных *i'*-го типа в последовательности $\pi^l$); *m*– количество данных, размещенных в последовательностях $\pi^l$ ($l = \overline{1,L}$) до текущего s-го шага алгоритма; $M_s$ – множество идентификаторов последовательностей $\pi^l$, в которых на данном текущем шаге алгоритма возможно изменение порядка выполнения программ (обработки данных); $N_s^p$ – множество направлений *g* изменения целевой функции *f* на *s*-ом шаге алгоритма, в которых гарантируется выполнение условия $^{-}\nabla f_g(s) \le 0$ для сформированных на основе исходного расписания $\pi(s)$ новых решений $\pi(s+g)$ ,определяемых путем изменения положения данных *i*-го типа в последовательностях $\pi^l$ ($l = \overline{1,L}$), количество последовательностей для одновременного изменения положения данных в них соответствует метрике текущей рассматриваемой окрестности $O(s)$; *I*– множество типов данных, которые должны быть обработаны в системе (типов данных, которые должны быть размещены в последовательностях $\pi^l$ ($l = \overline{1,L}$)).

Инициализация параметров алгоритма перед началом его реализации предполагает задание их значений следующим образом: $I = \{1,2,3,..,n\}$; *m=0*; $v^{\max} = 1$, *s* = 1, также задается значение максимальной метрики $O_{max}$. Алгоритм Greedy Sheduler определения эффективного местоположения данных *i*-го типа в последовательностях $\pi^l$ ($l = \overline{1,L}$) содержит следующие шаги.

1. Индекс типа данных *i'*, размещаемых в последовательностях $\pi^l$ ($l = \overline{1,L}$), определяется следующим образом: $i' = min\{i \mid i \in I\}$; номер типа данных $i'$ соответствует индексу (номеру) строки в матрицах $(P)^l$ и столбца в матрицах $(t_{ji}^{0l})$, значения в которых характеризуют данные этого типа.

2. Добавляемые в $\pi^l$ ($l = \overline{1,L}$) данные *i'*-го типа размещаются в концах этих последовательностей; для этого в $i'$-ой строке столбца $v^{\max}(s)$ матриц $(P)^l$ ($l = \overline{1,L}$) задаются значения, соответствующие этим данным (элементы $p^l_{i',v^{\max}} = 1$); действия с матрицами $P^l(s)$ на *s*-м шаге алгоритма, связанные с инициализацией элементов $v^{\max}$-х столбцов их $i'$-х строк, выполняются следующим образом: $p_{i',v^{\max}} = 1$, $p_{k,v^{\max}} = 0$, при $k = \overline{1,n}$ и $k \neq i'$.

3. Инициализация множества $M_s$ идентификаторов последовательностей $\pi^l$, в которых на текущем *s*-м шаге алгоритма возможно изменение порядка обработки данных выполняется следующим образом: $M_s = \{1,2,..,L\}$.

4. На основе матриц $P^l(s)$ ($l = \overline{1,L}$) реализуется вычисление элементов матриц $(t_{ji}^{0l}(s))$ ($l = \overline{1,L}$), матрицы $(t_{ji}^{0l}(s))$ вычисляются в порядке, начиная с минимального значения индекса *l* (при $l = \overline{1,L}$).

5. Для полученного вида матриц $P^l(s)$ и $(t_{ji}^{0l}(s))$ ($l = \overline{1,L}$) определяется значение критерия $f(s)$; сформированное таким образом решение в виде пар матриц ($P^l(s)$ ; $(t_{ji}^{0l}(s))$) при $l = \overline{1,L}$ фиксируется как локально эффективное $(((P^l(s)),(t_{ji}^{0l}(s)) \mid l = \overline{1,L})$*= $=(P^l(s);(t_{ji}^{0l}(s)) \mid l = \overline{1,L})$;

6. Индексам *v* текущих рассматриваемых столбцов матриц $(P^l(s))$ и строк матриц $(t_{ji}^{0l}(s))$ присваивается значение $v^{\max}(v = v^{\max})$; при выполнении условия $v^{\max}(s) = 1$ (добавляемые в $\pi^l$ ($l = \overline{1,L}$) данные занимают в них первую позицию) изменение порядка данных в $\pi^l$ не выполняется, реализуется переход к шагу 13.

7. Значение метрики окрестности $O(s)$ решения $\pi(s)$, в которой будет выполняться поиск локально эффективных решений, задается равным 1.

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

9. Изменение в последовательностях $\pi^l$ ($l \in M_s$) (при неизменном виде остальных последовательностей) порядка данных таким образом, что данные *i'*-го типа перемещаются на одну позицию в начало $\pi^l$ ($l \in M_s$); количество последовательностей $\pi^l$, в которых одновременно изменяется положение данных *i*-го типа, соответствует текущему значению окрестности $O(s)$; выполняемые действия с матрицами $(P^l(s))$ имеют вид $p^l_{i',v-1}(s+g) = 1$, $p^l_{i',v}(s+g) = 0$, $p^l_{i',v}(s+g) = 0$, $p^l_{k',v-1}(s+g) = 0$, $p^l_{k',v}(s+g) = 1$, где $k'$ – индексы строк в матрицах $(P^l(s))$, в которых элементы $p^l_{k',v-1}$ в *(v-1)*-м столбце равен 1 (на $s$-м шаге алгоритма – это предыдущая позиция для рассматриваемых данных, которую они занимают

на $(s+g)$-м шаге); в результате формируются матрицы $(P^l(s+g))$ ($l \in M_s$); индекс $v$ текущего рассматриваемого столбца, идентифицирующий местоположение в $\pi^l$ рассматриваемых данных, модифицируется: $v = v - 1$.

10. С использованием матриц $(P^l)$ (при $l = \overline{1,L}$) вычисляются матрицы $(t_{ji}^{0l})$ (при $l = \overline{1,L}$) – для каждого ($s$+$g$)-го шага алгоритма, а также значения критерия $f(s+g)$ (в итоге формируются решения $((P^l(s+q)),(t_{ji}^{0l}(s+q)) \mid l = \overline{1,L})$).

11. Для исходного решения $((P^l(s)),(t_{ji}^{0l}(s)) \mid l = \overline{1,L})$ и каждого решения $((P^l(s+g)),(t_{ji}^{0l}(s+g)) \mid l = \overline{1,L})$ вычисляются значения левых $\nabla_g^- f(s)$ либо правых $\nabla_g^+ f(s)$ дискретных градиентов целевой функции $f$ следующим образом [10]:

а) $\nabla_g^- f(s) = [f(s+g) - f(s)] \le 0$,

б) $\nabla_g^+ f(s) = [f(s+g) - f(s)] > 0$;

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

12. В множестве $N_s^p$ выбирается направление $g'$, для которого модуль отрицательного градиента $\nabla_g^- f(s) \le 0$ наибольший: $g' \to \max\limits_g (|\nabla_g^- f(s)|)$ при $\nabla_g^- f(s) \le 0$; полученное решение $((P^l(s+g')),(t_{ji}^{0l}(s+g')) \mid l = \overline{1,L})$ рассматривается как локально эффективное, на основе которого формируются последующие решения.

13. Полученное решение $((P^l(s+g')),(t_{ji}^{0l}(s+g')) \mid l = \overline{1,L})$ фиксируется как локально эффективное: $((P^l(s)),(t_{ji}^{0l}(s)) \mid l = \overline{1,L})^* = ((P^l(s+g')),(t_{ji}^{0l}(s+g')) \mid l = \overline{1,L})$, фиксируется значение номера шага алгоритма $s = s + g$; при выполнении условия $v \ge 1$ реализуется переход к шагу 7; при невыполнении условия $v \ge 1$ индексы $l$ последовательностей $\pi^l$ ($l \in M_s$), в которых на данном $s$-м шаге изменяется положение данных рассматриваемого $i'$-го типа исключаются из множества $M_s$: $M_s = M_s \setminus \{l\}$, где номера $l$ соответствуют группе последовательностей, идентифицируемой индексом $g$ (где $g$ – индекс шага алгоритма выполняемого относительно локально эффективного решения $\pi(s)$), при выполнении условия $M_s \ne \varnothing$ реализуется переход к шагу 7; при $M_s = \varnothing$ отсутствуют последовательности $\pi^l$ ($l \in M_s$), в которых может быть изменен порядок обработки данных, выполняется переход к шагу 16.

14. Если $N_s^p = \varnothing$, то в текущей окрестности с метрикой $O(s)$ локально эффективного решения $((P^l(s)),(t_{ji}^{0l}(s)) \mid l = \overline{1,L})^*$ (расписания $\pi^*(s)$) новое решение, уменьшающее значение целевой функции, не найдено, тогда текущее значение метрики модифицируется следующим образом: $O = O + 1$ – формируется модифицированное значение метрики $O(s)$, если $O(s) \le O_{max}$, то выполняется переход к шагу 8;

15. при выполнении условия $O(s) > O_{max}$ в максимальной окрестности $O_{max}$ текущего локально эффективного решения $((P^l(s)),(t_{ji}^{0l}(s)) \mid l = \overline{1,L})^*$ новое локально эффективное

решение не найдено, поэтому локально эффективное решение $((P^l(s)),(t_{ji}^{0l}(s))|l=\overline{1,L})$*, соответствующее порядку обработки данных с включенными в состав последовательностей $\pi^l$ $(l=\overline{1,L})$ данными $i'$-го типа, может быть проинтерпретировано как глобально эффективное решение для рассматриваемых типов данных: 1,2,.., $i'$; при реализации условия $O(s)>O_{max}$ выполняется переход к шагу 16.

16. Выполняется модификация множества *I* следующим образом: $I=I\setminus\{i'\}$, если для полученного в результате модификации множества *I* выполняется условие $I=\varnothing$, то данные всех *n* типов размещены в последовательностях $\pi^l$ $(l=\overline{1,L})$ эффективным образом, тогда должен быть выполнен переход на шаг 17; если $I\neq\varnothing$, то данные не всех типов размещены в последовательностях $\pi^l$ $(l=\overline{1,L})$, реализуется переход на шаг 1.

17. Останов алгоритма.

Анализ структуры алгоритма Greedy Sheduler показывает, что он реализует «жадный» подход к определению приближенных эффективных решений. Для определения возможности идентификации приближенных эффективных решений (расписаний выполнения программ обработки данных) доказана следующая теорема.

*Теорема.* Алгоритм Greedy Sheduler обладает свойствами жадных алгоритмов (является жадным алгоритмом) и обеспечивает формирование эффективных расписаний обработки данных.

*Доказательство*: Принцип жадного выбора предполагает, что решение, которое является локально оптимальным, может привести к глобальному оптимальному решению. При этом формирование решения обобщенной задачи оптимизации представляет собой последовательное решение подзадач, решение каждой последующей подзадачи определяется на основе локально эффективного решения предыдущей.

Начальное решение (начальное состояние последовательностей $\pi^l$ $(l=\overline{1,L})$, предполагающее размещение в них данных типа $i=1$) обозначим как $z_0$. Исходную рассматриваемую задачу, связанную с размещением в последовательностях $\pi^l$ $(l=\overline{1,L})$ данных, типы которых принадлежат множеству $N_1$ вида $N_1=\{2,3,4,..,n\}$, обозначим через $z_{n-1}$. Таким образом, задача $z_{n-1}$ состоит в размещении в $\pi^l$ $(l=\overline{1,L})$ данных, типы которых принадлежат $N_1$.

В исходной задаче $z_{n-1}$ может быть выделена подзадача $z'_1$, которая предполагает размещение в последовательностях данных $(i=2)$-го типа. Т.е. в задаче $z'_1$ реализуется жадный выбор с точки зрения определения эффективного местоположения данных $(i=2)$-го типа в последовательностях $\pi^l$. Обозначим решение задачи $z'_1$ как $rz'_1$. В результате выделения подзадачи $z'_1$ может быть сформирована подзадача $z_{n-2}$, связанная с размещением в $\pi^l$ $(l=\overline{1,L})$ данных типов, принадлежащих $N_2=\{3,4,..,n\}$. Новые подзадачи $z_{n-2}$ формулируются для множества данных $N_2$, полученного на основе множества $N_1$ путем исключения из $N_1$ (из исходной задачи $z_{n-1}$) данных $(i=2)$-го типа. При этом подзадача $z_{n-2}$, возникающая после жадного выбора (в задаче $z'_1$), аналогична исходной задаче $z_{n-1}$ (при этом $|N_1|>|N_2|$). Аналогично в подзадаче $z_{n-2}$ может быть выделена подзадача $z'_2$ размещения в $\pi^l$ $(l=\overline{1,L})$ данных $(i=3)$-го типа.

Обозначим решение подзадачи $z'_2$ как $rz'_2$, тогда решение $rz'_2$ формируется с использованием (на основе) решения $rz'_1$ задачи $z'_1$. При этом решение $rz'_2$ задачи $z'_2$ не вызывает изменения порядка обработки данных типов $i=1$ и $i=2$, полученного при решении задачи

$z'_1$. Таким образом, решение задачи $z_{n-2}$ формируется на основе решения $rz'_1$ задачи $z'_1$, которое является локально эффективным для данных типов $i=(1,2)$ (сформированным на основе жадного выбора). Тогда на основе задачи $z_{n-2}$ (с учетом выделения подзадачи $z'_2$) может быть сформирована задача $z_{n-3}$.

Рассуждая по аналогии, приходим к выводу, что размещение в последовательностях $\pi^l$ $(l=\overline{1,L})$ данных ($n-1$)-го типа (жадный выбор при получении локально эффективного решения $rz'_{n-2}$ задачи $z'_{n-2}$) позволяет сформировать (сформулировать) подзадачу $z_{n-(n-1)}=z_1$, решение которой обеспечит определение эффективного местоположения данных $n$-го типа в последовательностях $\pi^l$ $(l=\overline{1,L})$. Подзадаче $z_1$ может быть поставлена в соответствие подзадача $z'_{n-1}$, решение которой формируется на основе решения $rz'_{n-2}$ задачи $z'_{n-2}$. Таким образом, решение задачи определения расписания $\pi$ обработки данных *n* типов предполагает, что:

1) задача оптимизации приведена к виду, когда после сделанного жадного выбора по определению эффективного местоположения в последовательностях $\pi^l$ $(l=\overline{1,L})$ данных *i*-го типа необходимо решить только одну подзадачу $z_{n-i}$ по размещению в последовательностях $\pi^l$ данных, типы которых принадлежат множеству $N_i=\{i+1,i+2,..,n\}$;

2) решение исходной задачи $z_{n-1}$ (в итоге) трансформируется в решение подзадачи $z_1$ (связанной с размещением в $\pi^l$ $(l=\overline{1,L})$ данных *n*-го типа), которое должно быть получено на основе решения подзадачи $z_2$ (размещение в $\pi^l$ $(l=\overline{1,L})$ данных *(n-1)*-го типа) и т. д.; в свою очередь решение подзадачи $z_1$ формируется путем реализации жадного выбора, связанного с размещением данных *n*-го типа в последовательностях $\pi^l$ $(l=\overline{1,L})$;

3) решение подзадачи $z_{n-(i+1)}$, сформированной после выполненного жадного выбора в задаче в задаче $z'_i$ $(i=\overline{1,n-2})$, реализуется на основе решения $rz'_i$ и не предполагает при этом изменения порядка обработки данных, размещенных в последовательностях $\pi^l$ на предыдущих шагах алгоритма.

В итоге глобальное эффективное решение задачи $z_{n-1}$ по размещению в последовательностях $\pi^l$ $(l=\overline{1,L})$ данных (*n–1*)-го типа может быть получено из локальных эффективных решений подзадач $z'_1$, $z'_2$,.., $z'_{n-1}$. Любое решение $rz'_k$ подзадачи $z'_k$ содержит решения предыдущих подзадач $z'_1$, $z'_2$,.., $z'_{k-1}$, определяющих порядки обработки данных соответствующих типов в последовательностях $\pi^l$ $(l=\overline{1,L})$, для решения подзадачи $z'_{n-1}$ необходимо решить подзадачу $z'_{n-2}$ и т. д., для решения подзадачи $z'_2$ необходимо решить подзадачу $z'_1$. Или глобально эффективное решение $z_{n-1}$ может быть получено путем определения локально эффективных решений подзадач $z'_1$, $z'_2$,.., $z'_{n-1}$. При решении подзадачи $z'_k$ учитываются результаты подзадач $z'_1$, $z'_2$,.., $z'_{k-1}$, результаты последующих подзадач $z'_{k+1}$, ,.., $z'_{n-1}$ при этом не учитываются. С формулированные утверждения доказывают свойство (принцип) жадного выбора в сформированном алгоритме.

Рассуждение о том, что глобальное оптимальное эффективное решение формируется таким образом, что оно будет содержать решения всех подзадач позволяет сделать вывод о наличии структуры подзадач при определении глобального решения.

Таким образом, выполнение свойств жадного выбора для сформированного алгоритма является доказанным. Необходимо доказать возможность получения эффективных решений с его использованием.

Для этого выполним доказательство возможности представления рассматриваемой задачи (рассматриваемого объекта либо структуры подзадач) в виде матроида [10; 11]. Матроидом называется пара множеств $E, I$, состоящая из конечного множества $E$, называемого базовым множеством матроида, и множества его подмножеств $I$, называемого множеством независимых подмножеств матриода. Матроид $M = (E, I)$ обладает следующими свойствами [11]:

1) множество $I$ не пусто, если исходное множество $E$ является пустым ($E = \varnothing$), и будет состоять из одного элемента – $\varnothing$, т. е. $I = \varnothing$;

2) если $B \in I$, а $A \subseteq B$, то $A \in I$; т. е. если независимое множество $B$ входит в множество независимых подмножеств $I$, а $A$ является подмножеством $B$, то $A$ также входит в $I$;

3) если $A, B \in I$ и при этом $|A| < |B|$, то существует такой элемент *x* множества $B$, что $x \in (B \setminus A)$ и $A \cup \{x\} \in I$.

Таким образом, если для рассматриваемой задачи будут доказаны сформулированные свойства 1–3 матроида (т. е. задача (объект, система) обладает указанными свойствами 1–3 матроида), то ее решение может быть найдено с использованием жадного алгоритма.

Напомним, что множества *A* и *B* являются независимыми, если одно из них не является частью другого. Максимальное независимое множество $B \in I$ называется базой матроида.

С целью избежать громоздкости доказательства выполнимости рассмотренных условий для решаемой задачи опишем простейший случай при *L = 2* (рассматриваются два прибора – сегмента конвейера) и *n = 3* (рассматриваются данные трех типов). Для построения рассуждений расписание представим как совокупность наборов вида: $(i, j_1, j_2)$, где *i* – это тип данных, которые обрабатываются в системе; $j_1$ – это позиция данных *i*-го типа в последовательности $\pi^1$ на первом сегменте конвейера; $j_2$ – это позиция данных *i*-го типа типа в последовательности $\pi^2$ на втором сегменте конвейера. Тогда вид расписания обработки данных *i*-х типов ($i = \overline{1,3}$) на *L* сегментах конвейера ($L = 2$) следующий: $\pi = \{(1, j_1, j_2), (2, j_1, j_2), (3, j_1, j_2)\}$.

В этом случае два различных расписания $\pi$ и $\pi'$, определяющие порядок обработки данных в конвейерной системе, будут иметь вид $\pi = \{(1, j_1, j_2), (2, j_1, j_2), (3, j_1, j_2)\}$, $\pi' = \{(1, j'_1, j'_2), (2, j'_1, j'_2), (3, j'_1, j'_2)\}$.

Так как метод предполагает формирование различных вариантов решений, тогда $\pi \neq \pi'$. С учетом того, что независимость множеств определяется как невозможность для одного являться частью другого, тогда расписания $\pi$ и $\pi'$ (множества параметров вида $\{(i, j_1, j_2) \mid i = \overline{1,3}\}$) являются независимыми. Таким образом, так как $\pi \neq \pi'$ и $\pi \not\subseteq \pi'$ множество $\pi$ (расписание $\pi$) является независимым, так как $\pi' \not\subseteq \pi$, то множество $\pi'$ также является независимым. Соответственно, для рассматриваемых исходных данных множества $\pi$ и $\pi'$ являются максимальными, т. е. базами матроида *M*.

Допустим, что в множествах $\pi$ и $\pi'$ существуют такие наборы вида $(i, j_1, j_2)$, что $(i', j_1, j_2) = (i', j'_1, j'_2)$, где $i'$ – тип данных, имеющих одинаковые позиции на первом и втором приборах в расписаниях $\pi$ и $\pi'$. Тогда для реализации условия $\pi \neq \pi'$ необходимо, чтобы для остальных *i*-х типов данных выполнялось условие $(i, j_1, j_2) \neq (i, j'_1, j'_2)$ при $i \neq i'$. Тогда для подмножеств $\pi_1 = \{(i', j_1, j_2), (i, j_1, j_2) \mid i \neq i'\}$ и $\pi'_1 = \{(i', j'_1, j'_2), (i, j'_1, j'_2) \mid i \neq i'\}$ выполняется условие $\pi_1 \neq \pi'_1$ (для $i, i' \in \{1,2,3\}$ и $i \neq i'$). Очевидно, что для множеств $\pi_2$ и $\pi'_2$ вида $\pi_2 = \{(i, j_1, j_2) \mid i \neq i'\}$ и $\pi'_2 = \{(i, j'_1, j'_2) \mid i \neq i'\}$ условие $\pi_2 \neq \pi'_2$ выполняется автоматически.

Тогда $\pi_1 \subset \pi$, при этом множество $\pi \neq \pi'$, следовательно, в силу того, что $\pi_1 \neq \pi'_1$, имеем, что $\pi_1 \not\subseteq \pi'$. По аналогии $\pi'_1 \subset \pi'$, в силу того, что $\pi \neq \pi'$ и $\pi_1 \neq \pi'_1$, имеем, что $\pi'_1 \not\subseteq \pi$. Таким образом, подмножество $\pi_1$ является независимым по отношению к множеству $\pi'$, а подмножество $\pi'_1$ – независимо по отношению к множеству $\pi$. Тогда при $\pi_1 \subset \pi$ и $\pi'_1 \subset \pi'$ имеем, что $\pi_1 \in I$ и $\pi'_1 \in I$. Аналогичные рассуждения могут быть выполнены для подмножеств $\pi_2 \subset \pi$ и $\pi'_2 \subset \pi'$, тогда $\pi_2 \in I$ и $\pi'_2 \in I$. Второе свойство матроида применительно к рассматриваемой задаче может считаться доказанным.

Для доказательства третьего свойства определим независимые множества $\pi_1 \in I$ и $\pi_2 \in I$ в виде $\pi_1 = \{(i, j_{i1}, j_{i2}), (i', j_{i'1}, j_{i'2})\}$, $\pi_2 = \{(1, j_{11}, j_{12}), (2, j_{21}, j_{22}), (3, j_{31}, j_{32})\}$. В соответствии с условием 3 $|\pi_1| < |\pi_2|$ (где в $\pi_1$ индексы типов данных $i$, $i'$ могут принимать одно из значений 1, 2, 3). Тогда в множестве $\pi_2$ могут быть выделены подмножества $\pi_{21}, \pi_{22}, \pi_{23}$ следующего вида (мощностью 2):

$$\pi_{21} = \{(1, j_{11}, j_{12}), (2, j_{21}, j_{22})\},$$
$$\pi_{22} = \{(1, j_{11}, j_{12}), (3, j_{31}, j_{32})\},$$
$$\pi_{23} = \{(2, j_{21}, j_{22}), (3, j_{31}, j_{32})\}.$$

В силу того, что множества $\pi_1$ и $\pi_2$ независимы, имеем $\pi_1 \neq \pi_{21}; \pi_1 \neq \pi_{22}; \pi_1 \neq \pi_{23}$ (ситуация, когда либо $\pi_1 = \pi_{21}$, либо $\pi_1 = \pi_{22}$, либо $\pi_1 = \pi_{23}$ невозможна, так как $\pi_1$ и $\pi_2$ независимы). При условии $\pi_1 \neq \pi_{21}$ и $\pi_1 \neq \pi_{22}$ рассмотрим $i = 1$, тогда возможно либо $j_{i1} = j_{11}$ и $j_{i2} \neq j_{12}$, либо $j_{i1} \neq j_{11}$ и $j_{i2} = j_{12}$. В этом случае при $i' = 2$ возможными являются следующие ситуации: $j_{i'1} = j_{21}$ и $j_{i'2} \neq j_{22}$ (при $j_{i1} = j_{11}$ и $j_{i2} \neq j_{12}$) либо $j_{i'1} \neq j_{21}$ и $j_{i'2} = j_{12}$ (при $j_{i1} \neq j_{11}$ и $j_{i2} = j_{12}$). По аналогии при $i' = 3$ имеем $j_{i'1} = j_{31}$ и $j_{i'2} \neq j_{32}$ (при $j_{i1} = j_{11}$ и $j_{i2} \neq j_{12}$) либо $j_{i'1} \neq j_{31}$ и $j_{i'2} = j_{32}$ (при $j_{i1} \neq j_{11}$ и $j_{i2} = j_{12}$).

В общем виде приведенные условия могут быть представлены следующим образом: $j_{i'1} = j_{k1}$ и $j_{i'2} \neq j_{k2}$ (при $j_{i1} = j_{11}$ и $j_{i2} \neq j_{12}$) либо $j_{i'1} \neq j_{k1}$ и $j_{i'2} = j_{k2}$ (при $j_{i1} \neq j_{11}$ и $j_{i2} = j_{12}$), где $i' = k = 2$ либо $i' = k = 3$. Аналогичные рассуждения могут быть выполнены для случая, когда $i = 2$, а $i' = 1$ либо $i' = 3$, а также для случая, когда $i = 3$, а $i' = 1$ либо $i' = 2$.

Формирование нового набора $\pi'_1$ реализуется путем добавления в множество наборов параметров вида $\pi_1 = \{(i, j_{i1}, j_{i2}), (i', j_{i'1}, j_{i'2})\}$ нового набора вида $(i'', j_{i''1}, j_{i''2})$ (понятно, что $(i'', j_{i''1}, j_{i''2}) \in \pi_2$ и $(i'', j_{i''1}, j_{i''2}) \notin \pi_1$). Тогда добавление в $\pi_1$ данных $i''$-го типа (набора вида $(i'', j_{i''1}, j_{i''2})$) позволяет получить множество наборов параметров вида $\pi'_1 = \{(i, j_{i1}, j_{i2}), (i', j_{i'1}, j_{i'2}), (i'', j_{i''1}, j_{i''2})\}$, при этом исходная совокупность (множество) наборов параметров $\pi_1 \subset \pi'_1$ изменена не будет.

Так как $\pi_1 \neq \pi_{2k}$ (при $k = \overline{1,3}$), но $\pi_1 \subset \pi'_1$, при этом $\pi_{2k} \subset \pi_2$, тогда $\pi'_1 \neq \pi_2$ (и, следовательно, $\pi'_1 \not\subseteq \pi_2$), поэтому сформированное множества $\pi'_1$ и $\pi_2$ являются независимыми и, следовательно, $\pi'_1 \in I$ (т. е. $[\pi_1 \cup \{i'', j_{i''1}, j_{i''2}\}] \in I$), что и требовалось доказать. Это означает, что третье свойство матроидных структур для рассматриваемой задачи доказано и решаемая задача (система объектов) является матроидом.

В силу того, что применимость свойств матроида для рассматриваемой задачи является доказанной, необходимо определить возможность идентификации эффективного решения этой задачи с использованием жадного алгоритма. Для этого воспользуемся теоремой о применимости жадного алгоритма для матроида, приведенной в [11]. В соответствии с этой тео-

ремой в результате работы жадного алгоритма, примененного к взвешенному матроиду, получается подмножество эффективных решений.

Матроид называется взвешенным, если на множестве $E$ задана весовая функция $f$ со значениями на множестве положительных чисел. При этом функция $f$ является аддитивной и свойство аддитивности распространяется на все подмножества множества $E$. Вес подмножества определяется как сумма весов его элементов, т. е. $f(\pi) = \sum_{i=1}^{n} f_i$, где $f_i$ – простои сегментов конвейера, связанные с размещением в последовательностях $\pi^l$ ($l = \overline{1, L}$) данных $i$-го типа. В этом случае если первоначально решение $\pi$ имеет вид $\pi(1) = \{(1, j_1, j_2)\}$, то оно характеризуется значением суммарных простоев $L$ приборов при обработке данных ($i = 1$)-го типа. Значение функции $f(1)$ соответствует весу элемента $(1, j_1, j_2)$, который обозначен как $f_1$, это значение определяется на подмножестве $\{(1, j_1, j_2)\}$. Тогда при добавлении в расписание $\pi$ набора вида $(2, j_1, j_2)$ – формировании нового подмножества, входящего в $E$, получим следующий вид множества $\pi$: $\pi(2) = \{(1, j_1, j_2), (2, j_1, j_2)\}$. При этом добавление в $\pi$ набора параметров $(2, j_1, j_2)$ приводит к появлению простоев приборов, вызванных обработкой данных ($i = 2$)-го типа (вес, соответствующий этому набору параметров, обозначим как $f_2$). В этом случае значение $f(2)$ – это сумма весов $f_1$ и $f_2$, а само значение определяется на подмножестве элементов $\{(1, j_1, j_2), (2, j_1, j_2)\}$. Аналогичные рассуждения могут быть выполнены для набора параметров вида $(3, j_1, j_2)$. Значение $f(3)$, как сумма весов $f_1$, $f_2$, $f_3$, определяется на подмножестве $\{(1, j_1, j_2), (2, j_1, j_2), (3, j_1, j_2)\}$.

Таким образом, введенный в рассмотрение критерий эффективности расписаний, учитывающий простои оборудования (сегментов конвейера) является аддитивной весовой функцией, определенной на матроиде. Решаемая задача может быть представлена матроидом (удовлетворяет свойствам матроида) и для нее определена аддитивная весовая функция (аддитивная весовая функция определена на матроиде), тогда жадный алгоритм обеспечивает определение подмножества эффективных решений, в котором может быть выбрано одно эффективное решение. Тогда рассматриваемый алгоритм, обладающий свойствами жадного алгоритма, будет обеспечивать определение эффективного решения рассматриваемой задачи.

Выполненная программная реализация предложенного метода построения статических расписаний, позволила исследовать его эффективность в сравнении с другими методами. Результаты формирования последовательностей обработки данных в виде диаграмм Ганта для различного количества типов данных ($n$ = 5 и $n$ = 10) и различного числа обрабатывающих приборов (сегментов конвейера) представлены на рис. 1–2.

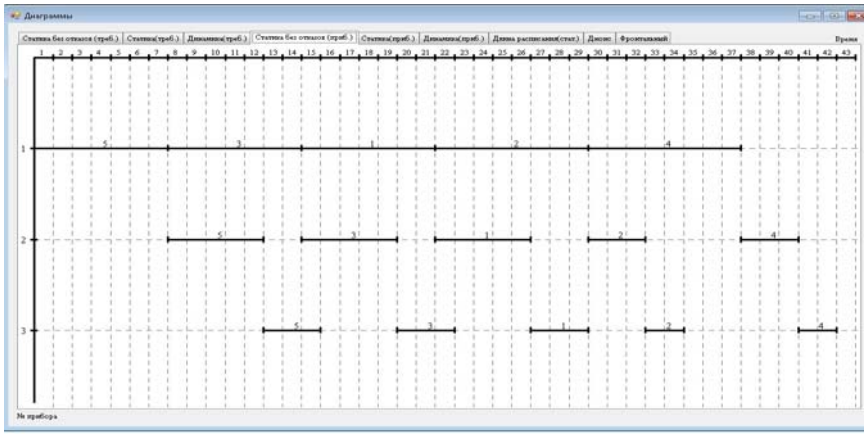

*Рис. 1.* Вид последовательностей выполнения программ обработки данных при $n$ = 5 и $L$ = 3

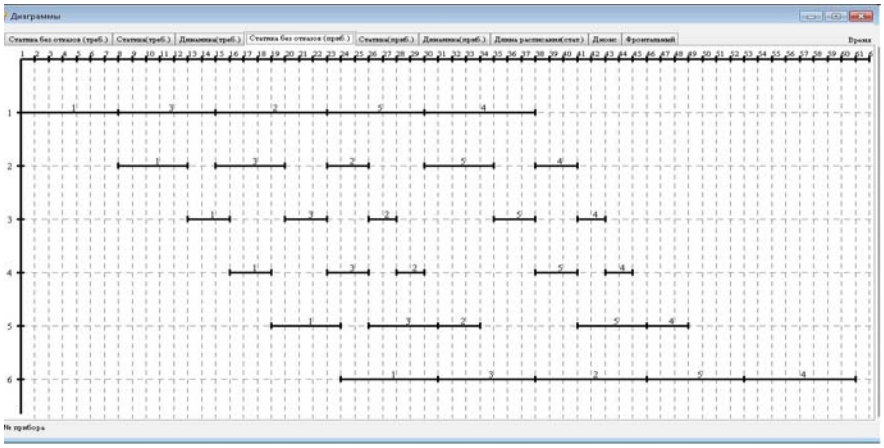

*Рис. 2.* Вид последовательностей выполнения программ обработки данных при $n = 5$ и $L = 6$

Идентификация эффективности разработанного алгоритма Greedy Sheduler выполнена путем сравнения результатов построения расписаний с результатами известных методов и альтернативных метаэвристических методов. В качестве задач для сравнения результатов использованы задача Джонсона для трех приборов [5], фронтальный метод с возрастанием длительностей обработки для данных в формируемых последовательностях [12], метод построения расписаний на основе генетических алгоритмов [13; 14] и метод отжига для построения расписаний [12]. Используемым критерием для задачи Джонсона и фронтального метода является значение момента времени окончания обработки всех данных в расписании (в введенных обозначениях этот критерий будет иметь вид $\max_{i,j}(\overline{t_{ji}^{L}})$. Для задачи Джонсона доказана [5] теорема об одинаковом виде последовательностей обработки данных на первом и втором приборах, а также на последнем и предпоследнем приборах. Поэтому в случае трех сегментов конвейера последовательности обработки данных на них будут одинаковыми. В связи с этим выполнено решение задачи построения расписаний обработки единичных данных для трех приборов (при задаваемом количестве типов данных $n$) с использованием подхода, предложенного при решении задачи Джонсона, а также фронтального метода, генетических алгоритмов и метода отжига. При этом в качестве критерия оптимизации, значение которого будет минимизировано, принят критерий вида $\max_{i,j}(\overline{t_{ji}^{L}})$. Полученные результаты исследований обобщены на графиках, представленных на рис. 3 (при этом использован относительный масштаб отображения результатов. Анализ результатов показал, что при малом количестве типов

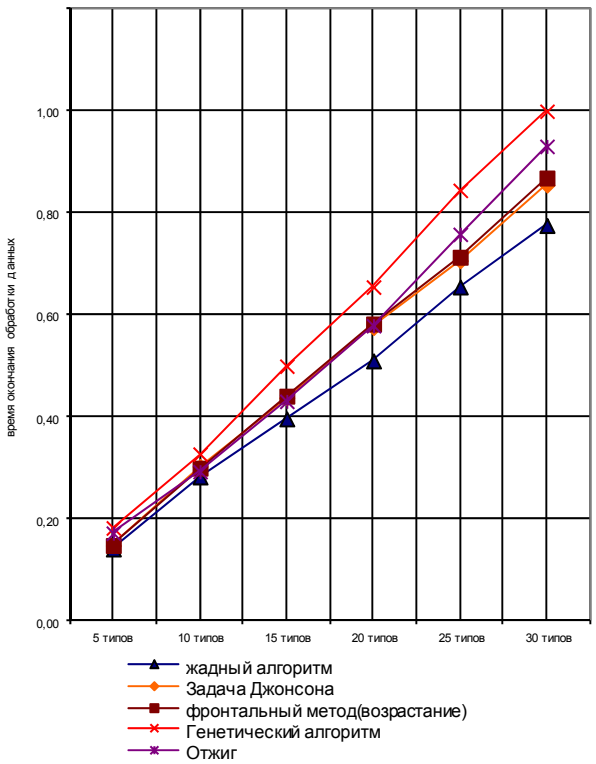


*Рис. 3* Сравнение эффективности методов построения расписаний для критерия $\max_{i,j}(\overline{t_{ji}^{L}})$

данных (5–15 типов единичных данных, обрабатываемых в системе) разработанный жадный алгоритм обеспечивает на 3–8 % более эффективные решения, при увеличении количества типов единичных данных до 20–30 разработанный жадный алгоритм обеспечивает на 10–15 % более эффективные решения, чем рассматриваемые методы. Также выполнено сравнение эффективности жадного алгоритма и метаэвристических методов (генетические алгоритмы и метод отжига), при этом для определения эффективных расписаний использован критерий, учитывающий простои сегментов конвейера при обработке данных, задаваемый выражением (4). Результаты сравнительных исследований эффективности метаэвристических методов и разработанного жадного алгоритма в относительном масштабе представлены на рис. 4.

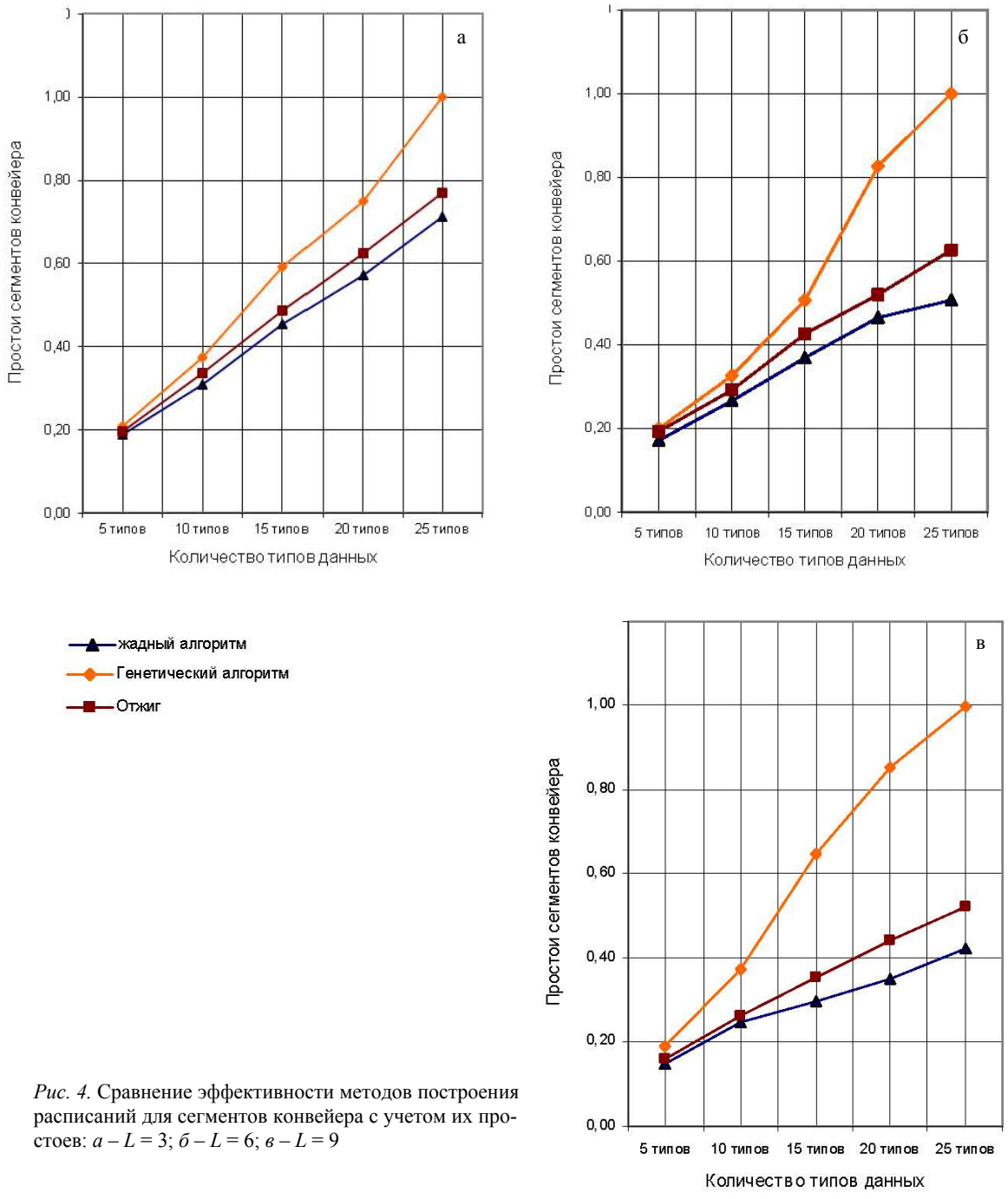


*Рис. 4.* Сравнение эффективности методов построения расписаний для сегментов конвейера с учетом их простоев: *а* – $L = 3$; *б* – $L = 6$; *в* – $L = 9$

Анализ результатов показал, что при разном количестве обрабатывающих приборов (трех, шести, девяти сегментах конвейера) для малого количества типов данных (5–15 типов единичных данных, обрабатываемых в системе) разработанный жадный алгоритм обеспечивает на 5–8 % более эффективные решения, при этом эффективность алгоритма более значительна для большего количества сегментов. При увеличении количества типов единичных данных до 20–30, разработанный жадный алгоритм обеспечивает на 10–15% более эффективные решения, чем рассматриваемые метаэвристические методы.

Выполненный анализ эффективности жадного алгоритма построения расписаний в сравнении с известными методами показал возможность его использования для получения решений с меньшими значениями критериев. Таким образом, разработанный метод позволяет определять эффективные решения по порядку обработки данных в конвейерных системах.

**Заключение**

Разработан алгоритм составления расписаний обработки единичных данных в конвейерных системах, который является более эффективным, чем известные метаэвристические алгоритмы. В то же время в отличие от задачи Джонсона и фронтального метода построения расписаний этот алгоритм позволяет определять порядок обработки данных для большего, чем 3, количества сегментов конвейера. Дальнейшие исследования должны быть направлены на разработку моделей и методов формирования динамических расписаний обработки единичных данных в конвейерной системе, в основу которых положен реализованный метод формирования статических расписаний обработки данных. При этом динамические свойства разрабатываемых алгоритмов вытекают из необходимости учета событий в системе, изменяющих запланированный ход вычислительного процесса.

**Список литературы**

1. *Хьюз К., Хьюз Т.* Параллельное и распределенное программирование на C++. М.: Вильямс, 2004. 672 с.

2. *Топорков В. В.* Модели распределенных вычислений. М.: ФИЗМАТЛИТ, 2004. 320 с.

3. *Воеводин В. В., Воеводин Вл. В.* Параллельные вычисления. СПб.: BHV– Петербург, 2002. 599 с.

4. *Сиротина Н. Ю.* Параллельные вычислительные системы. Красноярск: ФГОУ ВПО «Сибирский федеральный университет», 2008. 133 с.

5. *Лазарев А. А., Гафаров Е. Р.* Теория расписаний. Задачи и алгоритмы. М.: МГУ, 2011. 222 с.

6. *Лазарев А. А. Мусатова Е. Г., Кварцхелия А. Г., Гафаров Е. Р.* Теория расписаний. Задачи управления транспортными системами. М.: МГУ им. М. В. Ломоносова, 2012, 159 с.

7. *Долгова О. Э., Пересветов В. В.* Составление расписаний с минимизацией суммарного запаздывания на одном приборе методом параллельных муравьиных колоний. Вестн. ТОГУ Информатика, вычислительная техника и управление. 2012, №2 (25), с. 45–52.

8. *Pinedo M. L.* Sheduling.Theory,Algoritms, and Systems. Springer, 2008. 664 p.

9. *Ковалев М. М.* Модели и методы календарного планирования. Курс лекций. Минск: БГУ, 2004. 63 с.

10. *Ковалев М. М.* Матроиды в дискретной оптимизации. М.: Едиториал УРРС, 2003. 224 с.

11. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы. Построение и анализ. М.: МЦНМО, 2002. 960 с.

12. *Прилуцкий М. Х., Власов В. С.* Метод ветвей и границ с эвристическими оценками для конвейерной задачи теории расписаний. Математическое моделирование. Оптимальное управление. Вестн. Нижегород. ун-та им. Лобачевского Н. И., 2008, №3, с. 147–153.

13. *Рутковская Д., Пилиньский М., Рутковский Л.* Нейронные сети, генетические алгоритмы и нечеткие системы. М.: Горячая линия-Телеком, 2006. 284 с.

14. *Шаповалов Т. С.* Генетический алгоритм составления расписания запуска параллельных заданий в GRID. Многопроцессорные вычислительные системы, 2010, № 4 (26). С. 115–126.

*Материал поступил в редколлегию 16.12.2014*

**K. V. Krotov**

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

*krotov_k1@mail.ru*

## GRADIENT METHOD OF MAKING A STATIC SCHEDULES FOR CONVEYOR SYSTEMS BASED ON GREEDY STRATEGIES

The model of scheduling data in conveyor systems and method of constructing a static schedule, which based on greedy strategy, were found in work.

*Keywords*: multi-stage conveyor system; schedules performance data processing programs; gradient method; greedy algorithm.

**References**

1. K. Hughes. Parallel and Distributed Programming in C ++/ Hughes T. M .: Publishing House “Williams”, 2004. 672 p.

2. Toporkov V. V. Models of distributed computing. Moscow FIZMATLIT, 2004, 320 p.

3. Voevodin V. V., Voevodin Vl.V. Parallel computing. SPb, Publishing house «BHV-Petersburg», 2002. 599 p.

4. Sirotina N. Y. Parallel Computing System. Krasnoyarsk University FSEIHPE «Siberian Federal University», 2008, 133 p.

5. Lazarev A. A., Gafarov E.R. Scheduling theory. Problems and algorithms. Moscow, Publishing House of Moscow State University, 2011. 222 p.

6. Lazarev A. A., Musatov E. G., Kvartsheliya A. G., Gafarov E. R. Scheduling theory. Objectives of traffic management systems. Moscow, Publishing House of Moscow State University, 2012, 159 p.

7. Dolgov O. E., Peresvetov V. V. Scheduling minimizing total tardiness on a single device by parallel ant colony. / *Computer science*, *computer facilities and management*. *Bulletin PNU*, 2012, № 2 (25), p. 45-52.

8. Pinedo M. L. Sheduling.Theory,Algoritms, and Systems. Springer, 2008, 664 p.

9. Kovalev M. M. Models and methods of scheduling. Lectures. Minsk, Publishing house BSU, 2004, 63 p.Kovalev M. M. Matroids in discrete optimization. Moscow, Publishing house «Editorial IRDA», 2003. 224 p.

11. Kormen T., Leiserson C., Rivest R. Algorithms. Construction and analysis. Moscow, MTsNMO, 2002, 960 p. 16.

12. M. H. Prilutsky, V. S.Vlasov. Branch and bound method with heuristic estimates for conveyor scheduling problems. *Mathematical modeling. Optimal control. Bulletin of the Nizhny Novgorod University*, 2008, №3,. p.147-153.

13. Rutkovskaya D, Rutkowski J., Pilinsky M., Rutkovskiy L. M. Neural networks, genetic algorithms and fuzzy systems. Hotline Telecom, 2006, 284 p.

14. Shapovalov T. S. Genetic algorithm for scheduling run parallel jobs in GRID. *Multiprocessor computer systems*, 2010, №4 (26), p. 115-126.