УДК 004.421.2

А.А. Чусовлянкин, В.В. Морозенко

Национальный исследовательский университет «Высшая школа экономики», Пермь, Россия

ЛЕКСИКОГРАФИЧЕСКИЙ АЛГОРИТМ ДЛЯ РЕШЕНИЯ КОНВЕЙЕРНОЙ ЗАДАЧИ

Составление оптимальных расписаний, с одной стороны, является практической потребностью и диктуется необходимостью экономии ресурсов, например, времени выполнения комплекса заданий в многопроцессорных вычислительных системах. С другой стороны, многие из задач теории расписаний являются NP-трудными и не могут быть решены точно за полиномиальное время ни одним из известных алгоритмов. Конвейерная задача – это одна из известных задач, в которой несколько специализированных исполнителей должны в кратчайшие сроки выполнить набор многоэтапных заданий. Если все задания являются двухэтапными, то для решения конвейерной задачи существует точный алгоритм, имеющий полиномиальную сложность относительно числа заданий. Однако для случаев, когда число этапов превосходит 2, аналогичного алгоритма пока не существует. На практике в этой ситуации используют или тривиальный переборный алгоритм с экспоненциальной сложностью, когда число заданий не слишком велико, или какой-либо из быстрых приближенных алгоритмов, например фронтальный алгоритм. В данной работе предложен новый подход для быстрого приближенного решения конвейерной задачи с числом этапов, превосходящим два, а именно лексикографический алгоритм. В рассматриваемом алгоритме используется двойная сортировка. Сначала для каждого задания сортируются этапы по убыванию их длительности, в результате чего каждому заданию сопоставляется символьная строка, а затем полученные символьные строки сортируются по убыванию в лексикографическом порядке. Двойная сортировка способствует снижению вероятности возникновения интервалов вынужденного бездействия исполнителей. Подобные простои объясняются невозможностью выполнения последующего этапа какого-либо задания до тех пор, пока не завершится выполнение всех предыдущих этапов этого задания. Иногда простои неизбежны даже в оптимальных расписаниях. В таких случаях стремятся сократить время простоев. На наборе специально подготовленных тестовых заданий была изучена зависимость сложности, относительной погрешности и времени работы предложенного лексикографического алгоритма от количества заданий и числа этапов. Проведенное тестирование показало превосходство лексикографического алгоритма над фронтальным с точки зрения точности, а именно в большинстве случаев относительная погрешность лексикографического алгоритма оказалась ниже, чем у фронтального алгоритма, однако фронтальный алгоритм находил расписание быстрее.

Ключевые слова: теория расписаний, оптимальное расписание, конвейерная задача, приближенный алгоритм, погрешность алгоритма, лексикографический порядок, лексикографический алгоритм, фронтальный алгоритм.

A.A. Chusovliankin, V.V. Morozenko

National Research University «Higher School of Economics»,
Perm, Russian Federation

LEXICOGRAPHICAL ALGORITHM FOR SOLVING FLOW SHOP SCHEDULING PROBLEM

The optimal schedule, on the one hand, is a practical necessity to conserve resources, for example, problem in the multiprocessor computing systems. On the other hand, many of the scheduling problems are NP-hard and can not be solved exactly in polynomial time. Flow shop scheduling problem is the one of the most famous optimization problem. The scheduling problem is to find sequences of jobs on given machines with the objective of minimizing some function of the job completion times. All jobs pass through all machines in the same order. There is exact polynomial Johnson's algorithm for two machines. But it is NP-hard problem in case of an arbitrary number of machines. In practice, either trivial exhaustive search algorithms with exponential complexity are used when the number of jobs is not large, either some fast approximation algorithms, for example, the frontal algorithm. In this paper a new approach for the approximate solution of the flow shop scheduling problem is proposed. It uses double sorting. For each job, the stage durations are previously sorted in descending order. Stage numbers are written as a character string (classification). These job classifications lexicographically are sorted in descending order. Therefore, jobs with the longest last stage will be executed firstly. Then work with the longest penultimate stage will be executed and so on. It means that the last machine, and then the penultimate machine etc. at the beginning are loaded with the largest jobs. It allows them to avoid idle waiting for the next jobs. The practical implications of this paper is higher accuracy of the lexicographical algorithm compared with the frontal algorithm to works, where the number of jobs more than the number of stages, however the frontal algorithm is faster. The paper provides data about accuracy and execution time of these algorithms depending on the number of jobs and stages.

Keywords: scheduling theory, the optimum schedule, flow shop scheduling problem, approximate algorithm, algorithm accuracy, lexicographical order, lexicographical algorithm, frontal algorithm.

Введение. Теория расписаний – важная область прикладной математики, имеющая многочисленные и разнообразные применения [1, 2]. Потребность в составлении расписаний, оптимальных по какому-либо критерию, часто возникает на практике, например, при формировании последовательности исполняемых заданий в многопроцессорных вычислительных системах [3]. Правильный подбор такой последовательности позволяет экономить ресурсы – общее время выполнения, объем используемой памяти и задействованные вычислительные мощности.

Одной из проблем теории расписаний является так называемая *конвейерная задача*, которая, как известно, является NP-трудной для k-этапных заданий при $k > 2$ [4, 5]. Её решение для системы из n заданий с помощью полного перебора всех $n!$ возможных последовательностей выполнения работ при больших n практически не осуществимо, а точных полиномиальных алгоритмов для её решения до сих пор не найдено. По-

этому актуальной задачей является поиск эффективного алгоритма. Стоит отметить, что для частного случая конвейерной задачи, когда число этапов равно двум, существует точный полиномиальный *алгоритм Джонсона* [6]. Кроме того, для случая, когда число этапов $k > 2$, известен так называемый *фронтальный алгоритм*, относящийся к классу быстрых приближенных алгоритмов для решения конвейерной задачи [7].

В данной работе авторами предложен новый подход для решения конвейерной задачи, а именно *лексикографический алгоритм*, основанный на лексикографической сортировке символьных строк, сопоставленных по некоторому правилу каждому из заданий. Данные символьные строки содержат специальным образом структурированную информацию о длительности каждого этапа задания. Как показало тестирование, проведенное на большом наборе входных данных, предложенный алгоритм в подавляющем большинстве случаев имеет более высокую точность в сравнении с фронтальным алгоритмом.

Постановка задачи. Конвейерная задача заключается в следующем: пусть i – номер этапа задания (и номер исполнителя, $i = \overline{1,k}$), j – номер задания ($j = \overline{1,n}$), $T = \left\| t_{ij} \right\|$ – действительная матрица с положительными элементами размеров $k \times n$ (где k – количество этапов и исполнителей, n – количество заданий), у которой элемент t_{ij} определяет время выполнения i-го этапа j-го задания i-м исполнителем.

Требуется составить оптимальное расписание, т.е. найти последовательность выполнения всех заданий за минимальное время, удовлетворяющую следующим нескольким ограничениям:

1. Исполнитель не может приступить к выполнению своего этапа задания, пока не будут завершены все предыдущие этапы этого задания (поэтому задание не может выполняться одновременно несколькими исполнителями).

2. Один исполнитель не может одновременно выполнять несколько заданий [8].

Под временем выполнения всех заданий в данном случае понимается длительность временного промежутка с момента начала работы первого исполнителя до момента завершения работы k-го исполнителя.

Фронтальный алгоритм. Чтобы решить конвейерную задачу для набора заданий с числом этапов более двух, можно применить быстрый приближенный фронтальный алгоритм, который относится

к классу «жадных алгоритмов» – аналогу метода градиента для непрерывных задач [7].

Сначала для *j*-го задания находится величина его суммарной трудоемкости, $\tau_j = \sum_{i=1}^{k} t_{ij}$.

Задания выполняются в порядке невозрастания (либо неубывания) величин τ_j. Таким образом, предпочтение отдается заданиям с максимальной (либо минимальной) суммарной трудоемкостью. Далее будет приведен пример работы фронтального алгоритма на конкретном наборе из четырех трехэтапных заданий.

Лексикографический алгоритм. Авторами предлагается следующий лексикографический алгоритм:

1. Для каждого $j = \overline{1,n}$ выполняется сортировка длительностей t_{ij} всех этапов *j*-го задания в невозрастающем порядке. Номера этапов после сортировки записываются в виде символьной строки (классификации). Например, если трехэтапное задание *j* имеет длительности этапов $t_{1j} = 2,\ t_{2j} = 1,\ t_{3j} = 4,$ то это задание получает классификацию 3.1.2.

2. Полученные классификации заданий сортируются лексикографически в порядке убывания, поэтому в первую очередь будут выполнены задания, последние этапы которых имели наибольшую длительность по сравнению с длительностями остальных этапов этого же задания. Как будет показано далее, такая стратегия позволяет уменьшить потенциальные простои в ожидании возможности начать выполнение отдельных этапов для последующих заданий.

Ниже приведен псевдокод предлагаемого алгоритма.

```
    public class Work
    {
    public int id;              //номер задания
    public  int[] part;        //массив этапов (содержит
длительность)
    public string leks;     //лексикографический порядок
    }
    public static List<Work> Function(List<Work> works,
int n, int k)
    //n – количество заданий, k – количество этапов
    {
    foreach(Work w in works)
```

```
    {
    int[] mas = new int[n];
     //массив номеров этапов
         for (int i = 0; i < k; i++)
              mas[i] = i;
         Array.Sort(w.part, mas);
         //сортировка массива этапов по их длительности
в порядке возр.
         Array.Reverse(mas);
     //по убыванию (самый длинный этап вначале)
     w.leks = mas[0].ToString();
     //перевод массива в лексикограф. порядок
         for (int i = 1; i < k; i++)
                   {
                     w.leks += ".";
                     w.leks += mas[i].ToString();
                   }
    }
    List<Work> result = list.OrderByDescending(x =>
x.leks).
    ThenByDescending(x => x.part[x.leks[0] -
'0']).ToList();
    //сортировка заданий по убыванию лексикографическо-
го порядка, затем задания с одинаковым лексикографиче-
ским порядком сортируются в порядке убывания длительно-
сти максимального этапа
    return result;
    }
```

Заметим, что предлагаемый лексикографический алгоритм является приближенным, поскольку не всегда находит оптимальное расписание.

Исследование погрешности алгоритмов. Рассмотрим пример работы фронтального и лексикографического алгоритмов на конкретном наборе входных данных, состоящем из четырех трехэтапных заданий, т.е. $n = 4$, $k = 3$. Пусть задания имеют следующие длительности: первое задание – $t_{11} = 7$, $t_{21} = 1$, $t_{31} = 10$, второе задание – $t_{12} = 10$, $t_{22} = 10$, $t_{32} = 2$, третье задание – $t_{13} = 3$, $t_{23} = 4$, $t_{33} = 4$, четвертое задание – $t_{14} = 6$, $t_{24} = 3$, $t_{34} = 1$.

Фронтальный алгоритм в качестве ответа выдает последовательность выполнения заданий 4, 3, 1, 2, что соответствует расписанию

длительности 38 единиц и имеет относительную погрешность 12 % по сравнению с точным ответом. Данное решение изображено на рис. 1 в виде диаграммы Ганта, где указаны длительности заданий, а также моменты начала и окончания выполнения каждого из этапов [9].

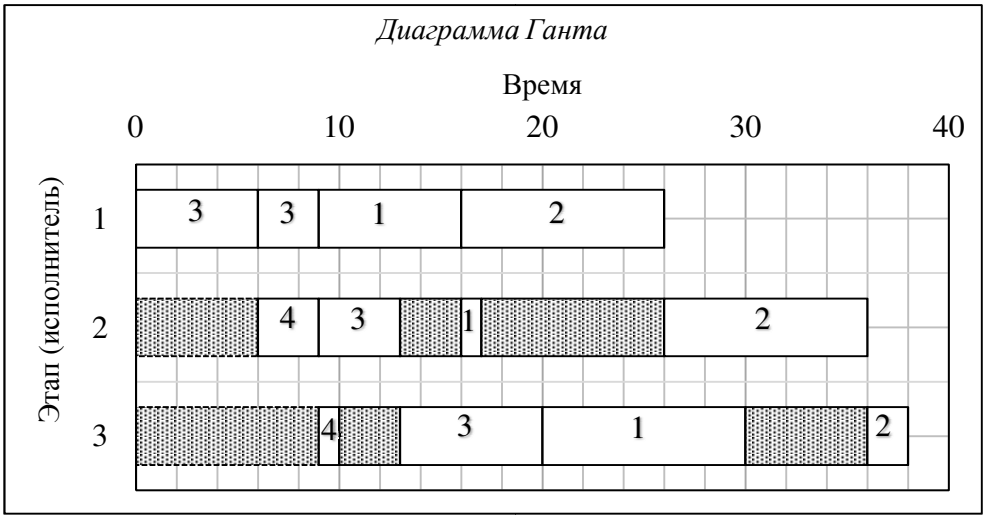

Рис. 1. Результат работы фронтального алгоритма

Предложенный лексикографический алгоритм предлагает иную последовательность выполнения заданий 3, 1, 2, 4, которая является правильной и представляет собой искомое оптимальное расписание, требующее для выполнения всего 34 единицы. Действительно, классификации заданий в данном примере представляют собой следующие символьные строки соответственно: 3.1.2, 1.2.3, 3.2.1, 1.2.3. Лексикографически наибольшей является строка 3.2.1, далее по убыванию идет строка 3.1.2, затем две одинаковых строки 1.2.3 и 1.2.3. Это соответствует последовательности выполнения заданий 3, 1, 2, 4 и расписанию, изображенному на рис. 2.

Для точного решения конвейерной задачи известен также метод ветвей и границ [10]. Для получения верхней оценки в этом методе можно, в частности, использовать приближенные алгоритмы. Однако для более качественной оценки, от которой существенно зависит скорость работы такого метода, требуются более точные алгоритмы, поэтому актуален поиск приближенных быстрых алгоритмов с высокой точностью [7]. Также на практике используются эвристические, метаэвристические и гибридные алгоритмы [11, 12]. К примеру, усовер-

шенствованный алгоритм дифференциальной эволюции решает конвейерную задачу с погрешностью около 3 % [13]. Приближенный итеративный алгоритм позволяет решать задачу со средней погрешностью 12 % [14].

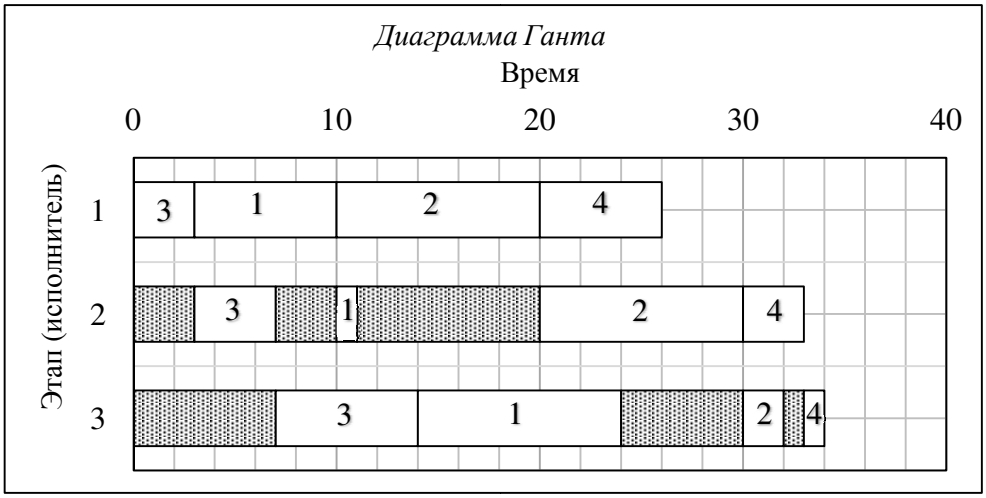

Рис. 2. Результат работы лексикографического алгоритма

Для сравнения погрешности лексикографического и фронтального алгоритмов разработаны программные модули на языке программирования C# и проведено исследование на ПК с тактовой частотой 1,7 GHz в среде разработке Visual Studio 2013. Длительности этапов случайным образом генерировались из диапазона [1, 50] при количестве заданий n в пределах от 2 до 9 и числе этапов k в пределах от 2 до 12.

На рис. 3 и 4 изображены соответственно графики зависимости относительной погрешности лексикографического и фронтального алгоритмов от количества этапов и числа заданий. Для каждого фиксированного количества этапов и числа заданий сгенерировано 100 000 тестов. Общее количество тестов равно 8 800 000.

На данных тестах лексикографический алгоритм демонстрирует лучшую точность по сравнению с фронтальным алгоритмом для набора тестов, где $n > k$, т.е. количество заданий больше числа исполнителей. Также стоит отметить, что для двухэтапных заданий средняя погрешность лексикографического алгоритма равна около 1 %, а максимальная полученная погрешность для лексикографического и фронтального алгоритмов составила 51 и 53 % соответственно.

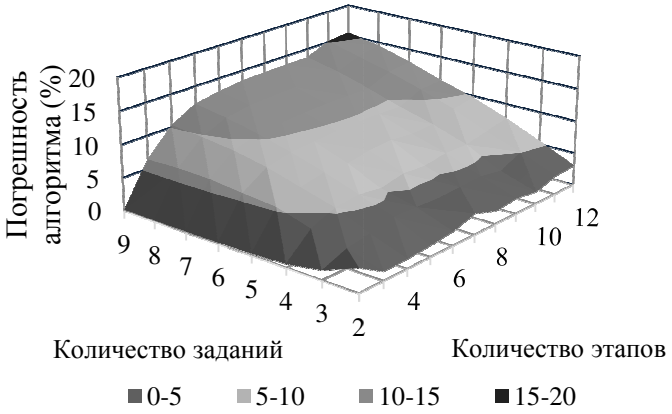

Рис. 3. Зависимость погрешности лексикографического алгоритма от количества заданий и этапов

Для большего количества заданий (100 заданий, количество этапов случайным образом генерировалось в диапазоне от 5 до 10) сгенерировано 100 000 тестов, среди которых в 81 % случаях ответ лексикографического алгоритма был точнее, чем фронтального алгоритма.

Временная сложность алгоритмов. Сложность лексикографического алгоритма зависит от сложности выбранного типа сортировки. К примеру, для сортировки со сложностью $O(n\log n)$ лексикографический алгоритм имеет следующую сложность:

$$O(nk(\log k + \log n)),$$

где k – количество этапов (и исполнителей), n – количество заданий. Такая сложность алгоритма обусловливается тем, что сначала необходимо отсортировать этапы для каждого задания, а затем и сами задания через лексикографическую сортировку соответствующих классификаций.

Для оценки временной сложности алгоритмов проведено исследование зависимости времени их работы от количества этапов (в пределах от 10 до 1000) и количества заданий (в пределах от 10 до 10 000). Для обоих алгоритмов рост количества этапов серьезнее влияет на время, чем количество заданий. Несмотря на это, для задачи большой размерности 1000×10 000 (количество этапов, количество заданий) время работы лексикографического алгоритма составляет примерно 6,5 мин, а время работы фронтального алгоритма – 3 с. Для сравнения стоит отметить, что метод ветвей и границ решает задачу размерности 5×10 за 51 с [7].

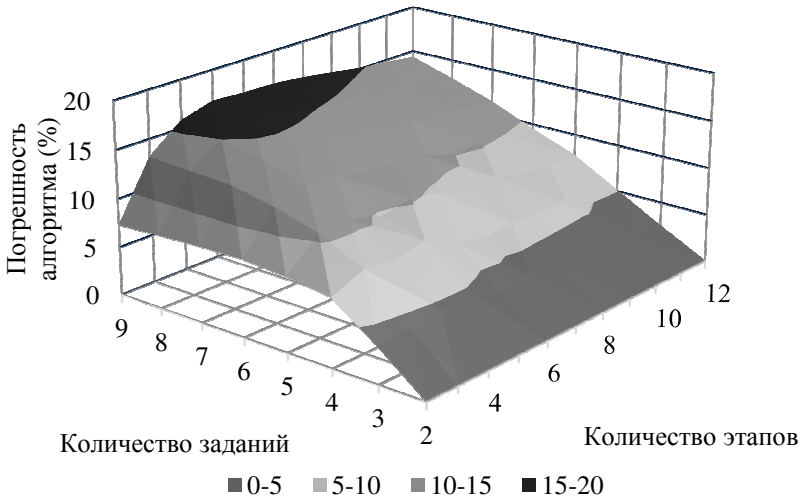

Рис. 4. Зависимость погрешности фронтального алгоритма от количества заданий и этапов

В табл. 1 и 2 продемонстрированы данные о времени работы (в секундах) лексикографического и фронтального алгоритма соответственно.

Таблица 1

Время работы лексикографического алгоритма (в секундах)

Количество этапов	Количество заданий			
	10	100	1000	10 000
10	0,00025	0,002	0,024	0,35
100	0,006	0,05	0,49	5
1000	0,35	3,45	35,6	389,7

Таблица 2

Время работы фронтального алгоритма (в секундах)

Количество этапов	Количество заданий			
	10	100	1000	10000
10	0,00004	0,0003	0,003	0,04
100	0,0002	0,002	0,02	0,23
1000	0,0014	0,015	0,25	2,8

Выводы. Проведенное исследование включало в себя разработку нового лексикографического алгоритма для решения конвейерной задачи с числом этапов более двух. Разработанный алгоритм был

реализован в виде программного продукта. Для его тестирования был подготовлен большой пакет тестовых заданий. Для заданий небольшой размерности точный ответ был предварительно получен с помощью программного модуля, реализующего тривиальный алгоритм, представляющий собой полный перебор всех возможных вариантов и требующий экспоненциального времени работы. Кроме того, для сравнения работы предложенного лексикографического и известного фронтального алгоритмов последний также был реализован в виде программного модуля.

Необходимо отметить, что предложенный авторами лексикографический алгоритм имеет простую внутреннюю логику, легко реализуем в виде программного продукта и может применяться на практике для приближенного решения задач большой размерности. Алгоритм имеет низкую относительную погрешность, в особенности для комплекса работ, где количество заданий больше, чем количество этапов. В большинстве случаев алгоритм находит более близкое к оптимальному решение, чем существующий фронтальный алгоритм. Погрешность лексикографического алгоритма и время его работы увеличиваются с увеличением количества заданий и количества этапов. Например, задачу размерности 1000×1000 алгоритм решает примерно за 35 с, что вполне приемлемо, если учесть, что единственный известный на данный момент точный алгоритм для решения конвейерной задачи такой размерности представляет собой полный перебор всех возможных вариантов, не осуществимый из-за неприемлемо значительных временных затрат.

Библиографический список

1. Коффман Э.Г. Теория расписаний и вычислительные машины. – М.: Наука, 1984. – 336 с.

2. Танаев В.С., Шкурба В.В. Введение в теорию расписаний. – М.: Наука, 1975. – 256 с.

3. Колесов Н.В., Толмачева М.В. Составление расписаний решения задач в конвейерных вычислительных системах // Информационно-управляющие системы. – 2005. – № 5. – С. 16–21.

4. Жданова Е.Г. Теория расписаний: учебник. – М.: Изд-во МГУ, 2000. – С. 61.

5. Pinedo M. Scheduling: Theory, Algorithms and Systems // Springer Science. – New York, 2008. – P. 152–172.

6. Garey M.R., Johnson D.S., Sethi R. The Complexity of Flowshop and Jobshop Scheduling // Mathematics of Operations Research. – 1976. – Vol. 1. – P. 117–129.

7. Прилуцкий М.Х., Власов В.С. Метод ветвей и границ с эвристическими оценками для конвейерной задачи теории расписаний // Вестник Нижегород. ун-та им. Н.И. Лобачевского. – 2008. – № 3. – С. 147–153.

8. Seda M. Mathematical Models of Flow Shop and Job Shop Scheduling Problems // World Academy of Science, Engineering and Technology, 2007. – P. 122–127.

9. Лазарев А.А., Гафаров Е.Р. Теория расписаний. Задачи и алгоритмы. – М.: Изд-во МГУ, 2011. – С. 19.

10. Brucker P. Scheduling Algorithms // Springer Verlag. – 2007. – P. 174–177.

11. Ruben Ruiz1, Jose Antonio Vazquez-Rodriguez The Hybrid Flow Shop Scheduling Problem // European Journal of Operational Research. – 2010. – Vol. 205, iss. 1. – P. 1–18.

12. Hariharan R., Golden Renjith Nimal R.J. Solving Flow Shop Scheduling Problems Using a Hybrid Genetic Scatter Search Algorithm // Middle-East Journal of Scientific Research. – 2014. – № 20(3). – P. 328–333.

13. Onwubolu G., Davendra D. Scheduling flow shops using differential evolution algorithm // European Journal of Operations Research. – 2006. – P. 674–679.

14. Канцедал С.А. Костикова М.В. Приближенный алгоритм для решения общей задачи теории расписаний с высокой точностью // Радиоэлектроника и информатика. – 1999. – № 4(9). – С. 97–105.

References

1. Koffman E.G. Teoriia raspisanii i vychislitel'nye mashiny [Scheduling theory and computer systems]. Moscow: Nauka, 1984. 336 p.

2. Tanaev V.S., Shkurba V.V. Vvedenie v teoriiu raspisanii [Scheduling theory introduction]. Moscow: Nauka, 1975. 256 p.

3. Kolesov N.V., Tolmacheva M.V. Sostavlenie raspisanii resheniia zadach v konveiernykh vychislitel'nykh sistemakh [Problem solving timetable

scheduling in assembly computer systems]. *Informatsionno-upravliaiushchie sistemy*, 2005, no. 5, pp. 16-21.

4. Zhdanova E.G. Teoriia raspisanii [Scheduling theory]. Moskovskii gosudarstvennyi universitet, 2000. p. 61.

5. Pinedo M. Scheduling: Theory, Algorithms and Systems. *Springer Science*. New York, 2008, pp. 152-172.

6. Garey M.R., Johnson D.S., Sethi R. The Complexity of Flowshop and Jobshop Scheduling. *Mathematics of Operations Research*, 1976, vol. 1, pp. 117-129.

7. Prilutskii M.Kh., Vlasov V.S. Metod vetvei i granits s evristicheskimi otsenkami dlia konveiernoi zadachi teorii raspisanii [Branch and bound method with heuristic evaluation for scheduling theory assembly tasks]. *Vestnik Nizhegorodckogo universiteta imeni N.I. Lobachevskogo*, 2008, no. 3, pp. 147-153.

8. Seda M. Mathematical Models of Flow Shop and Job Shop Scheduling Problems. *World Academy of Science, Engineering and Technology*, 2007, pp. 122-127.

9. Lazarev A.A., Gafarov E.R. Teoriia raspisanii. Zadachi i algoritmy. [Scheduling theory. Tasks and algorithms]. Moskovskii gosudarstvennyi universitet, 2011, p. 19.

10. Brucker P. Scheduling Algorithms. *Springer Verlag*, 2007, pp. 174-177.

11. Ruben Ruiz1, Jose Antonio Vazquez-Rodriguez The Hybrid Flow Shop Scheduling Problem. *European Journal of Operational Research*, 2010, vol. 205, iss. 1, pp. 1-18.

12. Hariharan R., Golden Renjith Nimal R.J. Solving Flow Shop Scheduling Problems Using a Hybrid Genetic Scatter Search Algorithm. *Middle-East Journal of Scientific Research*, 2014, no. 20(3), pp. 328-333.

13. Onwubolu G., Davendra D. Scheduling flow shops using differential evolution algorithm. *European Journal of Operations Research*, 2006, pp. 674-679.

14. Kantsedal S.A. Kostikova M.V. Priblizhennyi algoritm dlia resheniia obshchei zadachi teorii raspisanii s vysokoi tochnost'iu [Approximate algorithm for solving the general scheduling theory task with high accuracy]. *Radioelektronika i informatika*, 1999, no. 4(9), pp. 97-105.

Сведения об авторах

Чусовлянкин Алексей Александрович (Пермь, Россия) – студент Национального исследовательского университета «Высшая школа экономики» (614070, Пермь, ул. Студенческая, 38, e-mail: lixich@mail.ru).

Морозенко Владимир Викторович (Пермь, Россия) – кандидат физико-математических наук, доцент кафедры информационных технологий в бизнесе Национального исследовательского университета «Высшая школа экономики» (614070, Пермь, ул. Студенческая, 38, e-mail: v.morozenko@mail.ru).

About the authors

Chusovliankin Aleksei Aleksandrovich (Perm, Russian Federation) is a Student Higher School of Economics National Research University (614070, Perm, 38, Studencheskaya str., e-mail: lixich@mail.ru).

Morozenko Vladimir Viktorovich (Perm, Russian Federation) is a Ph.D. in Physical and Mathematical Sciences, Associate Professor at Department of Information Technologies in Business Higher School of Economics National Research University (614070, Perm, 38, Studencheskaya str., e-mail: v.morozenko@mail.ru).

Получено 12.10.2016