АЛГОРИТМИЗАЦИЯ ПОСТРОЕНИЯ РАСПИСАНИЙ, УЧИТЫВАЮЩИХ ВРЕМЕННЫЕ ОГРАНИЧЕНИЯ

УДК 004.02.021

Алексей Сергеевич Добрынин,
зав. лаб. кафедры автоматизации и информационных систем, старший преподаватель Сибирский государственный индустриальный университет (СибГИУ)
Тел.: 8 (3843) 78-43-76
Эл. почта: serpentfly@mail.ru

Роман Сергеевич Койнов,
зав. сектором кафедры АИС, старший преподаватель Сибирский государственный индустриальный университет (СибГИУ)
Тел.: 8 (3843) 78-43-76
Эл. почта: koynov_rs@mail.ru

Задачи построения расписаний (JSSP) в различных сферах человеческой деятельности имеют важнейшее теоретическое и прикладное значение. Во многих из них присутствуют специфические временные ограничения, описывающие допустимые для планирования отрезки времени и периоды простоев на интервале планирования. Статья описывает алгоритм распределения работ в условиях временных ограничений для задач построения производственных и учебных расписаний, а также сервисной деятельности.

Ключевые слова: *распределение работ, расписание, ИТ-сервис, планирование работ, временные ограничения, сервисное управление*

Alexey S. Dobrynin,
Head of the Laboratory of the Department of Automatization and informational systems, Senior Lecturer Siberian State Industrial University (SibSIU)
Tel.: 8 (3843) 78-43-76
E-mail: serpentfly@mail.ru

Roman S. Koynov,
Head of sector of the Department of Automatization and informational systems, Senior Lecturer Siberian State Industrial University (SibSIU)
Tel.: 8 (3843) 78-43-76
E-mail: koynov_rs@mail.ru

ALGORITHMIC CONSTRUCTION SCHEDULES IN CONDITIONS OF TIMING CONSTRAINTS

Tasks of time-schedule construction (JSSP) in various fields of human activities have an important theoretical and practical significance. The main feature of these tasks is a timing requirement, describing allowed planning time periods and periods of downtime. This article describes implementation variations of the work scheduling algorithm under timing requirements for the tasks of industrial time-schedules construction, and service activities.

Keywords: *labeling, timetable, it-service, job shop schedule, time constraints, service management*

1. Введение

В статье излагаются алгоритм построения расписаний планирования работ, операций в условиях временных ограничений. Данная задача актуальна для промышленных предприятий, организаций различных сфер деятельности, в которых присутствует временное планирование.

Рассматриваемая задача базируется на классической задаче назначения работ (сетевое планирование – JSSP) для непрерывного времени. Постановка рассматриваемой в статье задачи опирается на решение, полученное для непрерывного времени и набора временных ограничений.

Поскольку большинство организаций осуществляет деятельность, опираясь на рабочие графики, алгоритм, рассматриваемый в данной публикации, позволяет получить более точные решения для реальных производственных систем, в которых присутствуют ограничения.

2. Элементы математической модели

Базовая задача построения производственных расписаний [1], для непрерывного времени, формализуется как задача на графах, в которой узлы представляют собой события, дуги – отдельные процессы или работы. С каждой дугой ассоциирован двухкомпонентный вес, представленный вещественным числом и временной разницей с возможностью их взаимного отождествления. Этапы решения базовой задачи [1], реализованы в рамках модельно – алгоритмического комплекса (МАК) [2] и дают неплохие результаты на практике.

Особый интерес представляет задача, в которой необходимо учитывать ограничения, связанные с невозможностью распределить работы в определенный интервал времени. Сложность заключается в вариативном характере таких ограничений, которые могут изменяться в различных постановках. Рассмотрим элементы математической модели для достаточно общего случая, предполагая, что на периодических интервалах времени $t + \Delta t$ структура ограничений одинакова.

Одним из элементов математической модели, используемой для построения расписаний в ограничениях, является ***вектор кортежей работ*** $\overline{W}$, полученный в ходе решения задачи [1], где каждая отдельная запись представляет собой параметры отдельной работы, такие как: идентификатор работы (ID), дата начала (beginDate), дата раннего окончания (earlyEndDate), дата позднего окончания (lastEndDate), компонент временного смещения (offsetDate).

$$w_i = \{D_i, beginDate_i, earlyEndDate_i, lastEndDate_i, offsetDate_i, \} \qquad (1)$$

С точки зрения процедуры составления расписаний, отдельный кортеж (запись) представляет набор связанных данных, по отношению к некоторому идентификатору работы, часть из которых используется алгоритмом построения расписаний.

Также важнейшим элементом математической модели является логическая ***матрица работ и простоев*** $timeMap[d \in Days, h \in Hours]$, которая описывает временную сетку интервалов проведения работ, такую что:

$$timeMap[d, h] = \begin{cases} 1, & \textit{допустимо размещение элемента работы} \\ 0, & \textit{простой, размещение не допускается} \end{cases} \qquad (2)$$

Для случаев описания детальных временных компонент, матрица работ и простоев может быть трансформирована в ***кортеж работ и простоев*** (использование более двух временных компонент). В общем случае, структура и вид матрицы или кортежа зависит от размерности времени, требуемой точности задания отрезков времени и динамики процессов. В задачах

построения производственных расписаний целесообразно использовать «сжатую» интерпретацию, когда известно, что производственные процессы четко привязаны к конкретным дням недели, см. выражение (3).

$$M[DayOfWeek[d], h] = \begin{cases} 1, & \text{допустимо размещение элемента работы} \\ 0, & \text{простой, размещение не допускается} \end{cases} \quad (3)$$

Введем понятие левого и правого ***временного сдвига***, которое будет означать единичное приращение минимальной компоненты кортежа в сторону уменьшения или увеличения времени. Таким образом, для кортежа $K[d \notin Day, h \notin Hour, m \notin Minute,]$ сдвигом будет кортеж $K[d, h, m \pm 1]$. Рассматриваемый в статье алгоритм назначения работ (time – labeling) использует модель ограничений, представленных выражением 2.

3. Ключевые идеи алгоритма

Для упрощения понимания сути работы алгоритма в целом, выделим несколько ключевых идей:

1) *Двухкомпонентный, двунаправленный временной итерационный механизм.*

Итератор workIterator сдвигает временной кортеж в прямом направлении, итератор durationIterator сдвигает временной кортеж в обратном направлении, см. рисунок 1.

Механизм сдвига, с учетом смещения, непосредственно влияющий на окрестность работ $\overline{W}_N \notin \overline{W}$, *расположенных справа относительно текущей работы* w_i.

Возможность поиска работ, расположенных в окрестности текущей w_i справа или слева, достигается за счет реализации в информационной модели дуги графа ссылок на стартовый и конечный узел, см. Листинг 1.

Информационная модель дуги графа содержит ссылки на стартовый и конечный узел графа, реализующие поведенческий механизм *INode <T>*, также имеется возможность работы по уникальным строковым идентификаторам узлов. Таким образом, итерационный процесс по отдельной дуге графа воздействует на окрестность дуг, расположенных после текущей дуги, см. рисунок 2.

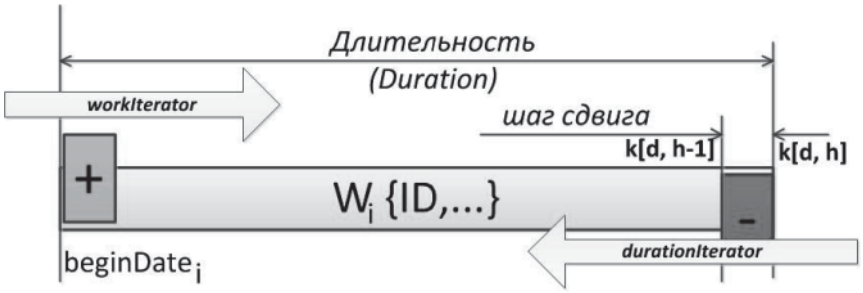

Рис. 1. *Двунаправленная итерация по времени*

```
/*Интерфейс для дуги графа IEdge<T>**/
public interface IEdge<T> : IComparable<IEdge<T>>
{
    ///Временной интервал дуги (выраженный через временную разницу)
    TimeSpan Duration { get; set; }
    //// Стартовый узел для дуги, как INode<T>
    INode<T> start_node { get; }
    //// Конечный узел дуги, как INode<T>
    INode<T> end_node { get; }

    string start_nodeid { get; } ///Стартовый идентификатор узла
    string end_nodeid { get; }   ///Конечный идентификатор узла
    string name { get; set; }    ///Наименование дуги
    string manager { get; set; }
    double Weigth { get; set; }  ///Вес дуги
}
```

Листинг 1. *Информационная модель дуги графа*

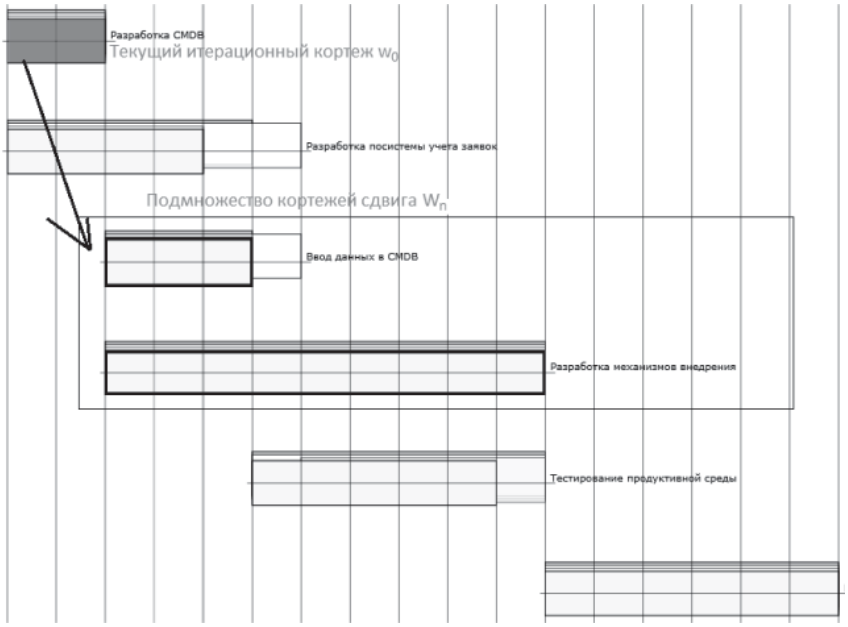

Рис. 2. *Подмножества сдвига при итерационном движении*

2) *Оригинальный механизм временной разметки (time-labeling) с использованием списка запретов.*

Суть итерационного механизма заключается в следующем: если на очередном i-шаге итерации элемент кортежа $k_i[d, h]$ для работы w_i не может быть распределен, происходит сдвиг всех временных характеристик работ окрестности справа w_i с

учетом смещения для следующей работы, на интервал времени $k_i[d, h + 1]$, если он отсутствует в списке запретов. В противном случае длительность текущей работы уменьшается на интервал времени $k_i[d, h - 1]$, при этом сдвига временных характеристик работ окрестности справа w_i не происходит. Так как имеется n-работ окрестности слева, для текущей работы w_i итерирование каждой из которых приводит к сдвигам w_i, целесообразно использовать список запретов *tabooList*, каждый элемент которого представляет собой кортеж $k_{taboo}[id, k_i[d, h]]$. Список запретов создается отдельно для каждой работы w_i и содержит даты, которые уже использовались ранее для сдвига работы w_i.

4. Содержательное описание алгоритма

Опираясь на описанные выше идеи, сформулируем алгоритм построения расписаний, пригодный для построения расписаний для сколь угодно сложных практических случаев временных ограничений, при условии их однородности.

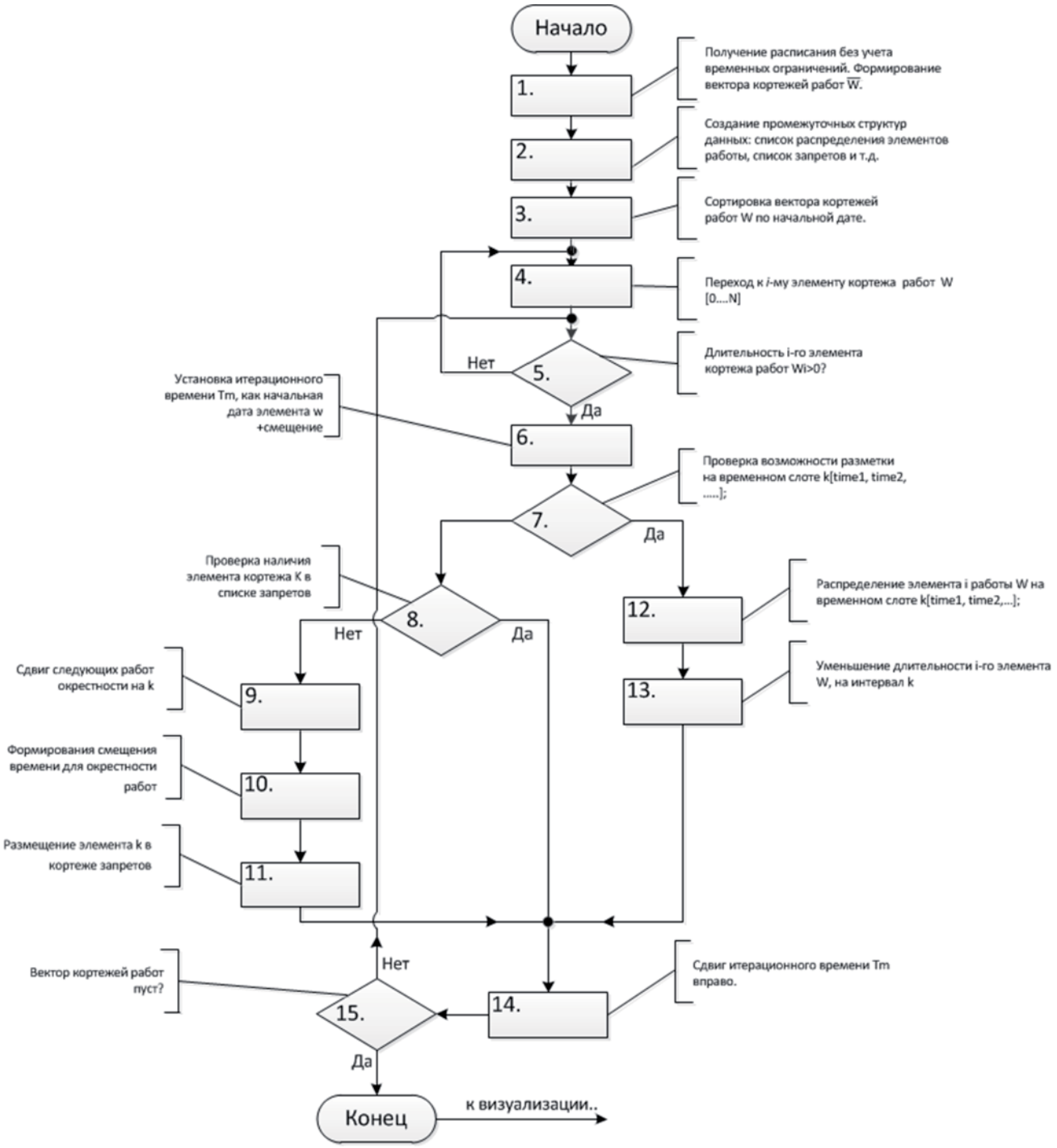

Рис. 3. *Блок-схема алгоритма разметки (labeling) работ*

1) Сортировка вектора w_i по возрастанию даты начала работы *beginDate*.

2) Определение даты начала проекта *prjDate* как $w_0\{...,$ *beginDate*,...$\}$.

3) Двунаправленная итерационная процедура по каждой работе $w_i \notin \overline{W}, i = 0...(N-1)$, выполняем действия по формированию кортежей ее размещения во времени. Формирование вектора кортежей $\overline{LBL}$, каждый элемент которого содержит идентификатор работы и дату начала разметки для временного сдвига.

Визуализация вектора кортежей $\overline{LBL}$ с использованием механизма рендеринга WPF.

5. Блок схема алгоритма

Представленная в данном разделе блок – схема алгоритма построена с опорой на процесс отладки работающей реализации на языке программирования C# в рамках модельно-алгоритмического комплекса (МАК) построения расписаний [2]. Алгоритм был опробован на 10 тестовых структурах графов, при произвольной генерации значений для матрицы временных ограничений. Блок схема представлена на рисунке 3.

6. Заключение

Рассмотренный в работе алгоритм реализован в составе модельно – алгоритмического комплекса МАК[2], опробован на множестве модельных структур графов работ (более 10), при произвольных способах задания кортежей временных ограничений.

Литература

1. Добрынин А.С., Кулаков С.М., Зимин В.В. Формализация задачи составления расписаний для стадии внедрения ИТ-сервиса // Научное обозрение: теория и практика. – 2013. – №2. – С. 47–52, 110.

2. О формировании комплекса инструментальных средств ИТ-провайдера для построения расписаний процесса внедрения сервиса / А.С. Добрынин, С.М. Кулаков, В.В. Зимин, Н.Ф. Бондарь // Научное обозрение. – 2013. – №8. – С. 93–101.

3. Р.С. Мартин, М. Мартин. Принципы, паттерны и методики быстрой разработки приложений на языке программирования C#. – М.: Символ-Плюс, 2013. – 786 с.

4. OGC-ITIL V3-2 Service Transition, TSO. – 2007.

References

1. Dobrynin A.S., Kulakov S.M., Zimin V.V. Formalization of a problem of drawing up schedules for a stage of introduction of IT service // Nauchnoe obozrenie: teoriya i praktika. – 2013. – №2. – S. 47–52, 110.

2. About formation of a complex of tools of IT provider for creation of schedules of process of introduction of service / A.S. Dobrynin, S.M. Kulakov, V.V. Zimin, N.F. Bondar' // Nauchnoe obozrenie. – 2013. – №8. – S. 93–101.

3. R.S. Martin, M. Martin. The principles, patterns and techniques of fast applications programming in the C# programming language. – M.:Simvol-Pljus, 2013. – 786 p.

4. OGC-ITIL V3-2 Service Transition, TSO. – 2007.