ДОБРЫНИН Алексей Сергеевич -
старший преподаватель кафедры АИС
ФБГОУ ВПО «Сибирский государственный
индустриальный университет»
(ФБГОУ ВПО «СибГИУ»)
Адрес: 654007, г. Новокузнецк, ул. Кирова, 42
e-mail: mail@sa.sibsin.ru

КОЙНОВ Роман Сергеевич -
зав. сектором информационного обеспечения
кафедры АИС ФБГОУ ВПО «СибГИУ»
Адрес: 654007, г. Новокузнецк, ул. Кирова, 42
e-mail: koynov_rs@mail.ru

АЛГОРИТМИЗАЦИЯ ПОСТРОЕНИЯ РАСПИСАНИЙ, УЧИТЫВАЮЩИХ ВРЕМЕННЫЕ ОГРАНИЧЕНИЯ

Введение

Статья рассматривает вариант реализации алгоритма распределения работ в условиях временных ограничений. В работе [2] рассматривались вопросы реализации модельно-алгоритмического комплекса при построении расписаний планирования работ и операций для непрерывного времени. Создание конструктора графов, реализация базовых алгоритмов поиска путей и визуализации графов позволяет решать более сложные задачи. Рассматриваемый в статье алгоритм предполагает, что задача сетевого планирования для непрерывного времени [1] уже решена на компьютере [2] (технология WPF - C#) для любых, сколь угодно сложных, структур орграфа. Однако практические требования чаще всего предполагают невозможность планирования на непрерывных временных интервалах. Длительность смены рабочих коллективов ограничена, деятельность людей на производстве осуществляется в определенные моменты времени. Необходимость точного планирования, учета графиков рабочего времени людей и других факторов, таких как загруженность эксплуатационной среды, приводят к появлению временных ограничений в постановке задачи временного планирования. Вариант реализации алгоритма построения расписаний (графиков работ) для временных ограничений произвольного вида рассматривается здесь.

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 выполнять работы в определенные отрезки 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 компонентов математической модели объекта планирования, используемой для построения расписаний в ограничениях, является ***вектор кортежей работ*** $\overline{W}$, полученный в ходе решения задачи [1], где каждая отдельная запись представляет со-

бой параметры отдельной работы, такие как: идентификатор работы (ID), дата начала (beginDate), дата раннего окончания (earlyEndDate), дата позднего окончания (lastEndDate), компонент временного смещения (offsetDate).

$$w_i = \{ID_i, beginDate_i, earlyEndDate_i, lastEndDate_i, offsetDate_i\} \quad (1).$$

В рамках процедуры составления расписаний отдельный кортеж (запись) представляет набор связанных данных, привязанных к некоторому идентификатору работы, часть из которых используется алгоритмом построения расписаний.

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

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

При детальном описании временных компонент матрица работ и простоев может быть трансформирована в ***кортеж работ и простоев*** (при наличии более двух временных компонент). В общем случае структура 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:

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

Введем понятие прямого и обратного ***временного сдвига***, которое будет означать единичное приращение минимальной компоненты кортежа в сторону уменьшения или увеличения времени. Например, для кортежа $K[d \in Day, h \in Hour, m \in Minute]$ сдвигом будет кортеж $K[d, h, m + /\text{-}1]$. Рассматриваемый в статье алгоритм назначения работ (time - labeling) использует модель ограничений, представленных выражением 3.

Ключевые механизмы алгоритма

Для упрощения понимания сути алгоритма выделим несколько ключевых механизмов:

1) Двухкомпонентный, двунаправленный механизм временных итераций (МВИ).

Итератор workIterator сдвигает временной кортеж в прямом направлении, итератор durationIterator сдвигает временной кортеж в обратном направлении (**рис. 1**).

2) Механизм сдвига (МС), который учитывает смещения и непосредственно влияет на окрестность $\overline{W}_N \in \overline{W}$ работ, расположенных справа относительно текущей работы w_i.

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

```
/*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*

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

3) Механизм временной разметки - MBP, (time-labeling) с использованием списка запретов.

Суть итерационного механизма заключается в следующем: если на очередном in-м шаге итерации элемент кортежа $k_i[d, h]$ для работы w_i не может быть распределен, происходит сдвиг всех временных характеристик работ окрестности справа от w_i с учетом смещения для следующей работы на интервал времени $k_i[d, h+1]$, если таковой отсутствует в списке запретов. В противном случае длительность текущей работы уменьшается на интервал времени $k_i[d, h-1]$, при этом сдвиг временных характеристик работ окрестно-

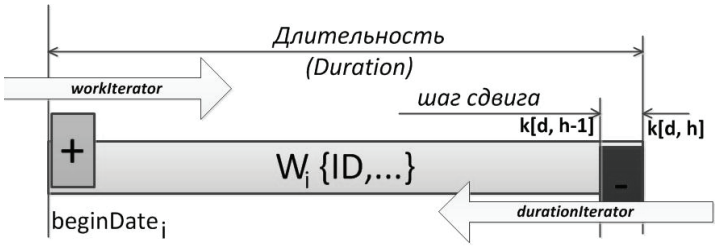

Рис. 1. *Схема двунаправленных итераций по времени*

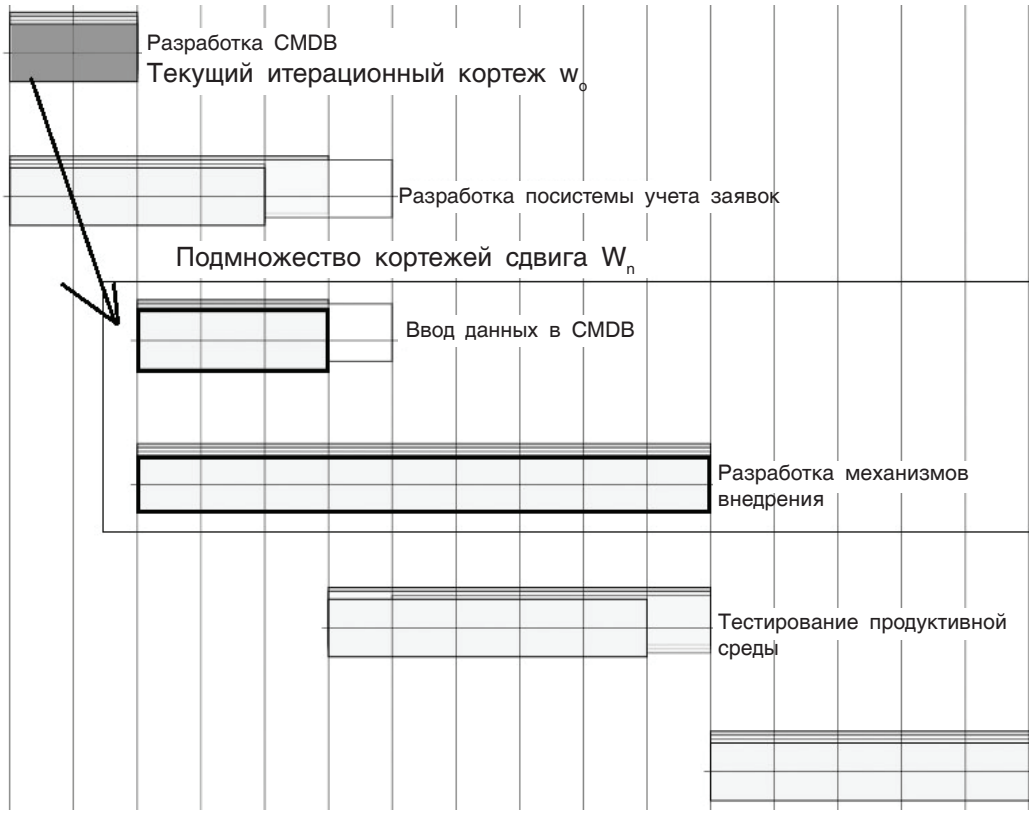

Рис. 2. *Подмножества сдвига при итерационном движении*

сти справа w_i не производится. Так как имеется n работ в окрестности слева от текущей работы w_i и итерирование каждой из них 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.

Содержательное описание алгоритма

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

1) Сортировка вектора $\overline{W}$ по возрастанию даты начала работы *beginDate*.

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

3) Двунаправленная итерационная процедура по каждой работе $w_i \in \overline{W}$, $i = 0...N-1$, включающая 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.

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

Блок-схема алгоритма

Представленная в данном разделе блок-схема построена с опорой на процесс отладки работающей версии 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 **рис. 3**.

Заключение

В ходе проведенного исследования был разработан и реализован алгоритм построения расписаний для задач календарного планирования работ, учитывающий временные ограничения. Результаты исследования закреплены авторскими свидетельствами и правами на интеллектуальную собственность. Практическая значимость заключается в возможности применения результатов исследования в следующих сферах деятельности:

1) управление проектами и планирование;

2) построение расписаний работ крупных промышленных комплексов и технологических агрегатов;

3) ресурсное-ролевое и рисковое управление.

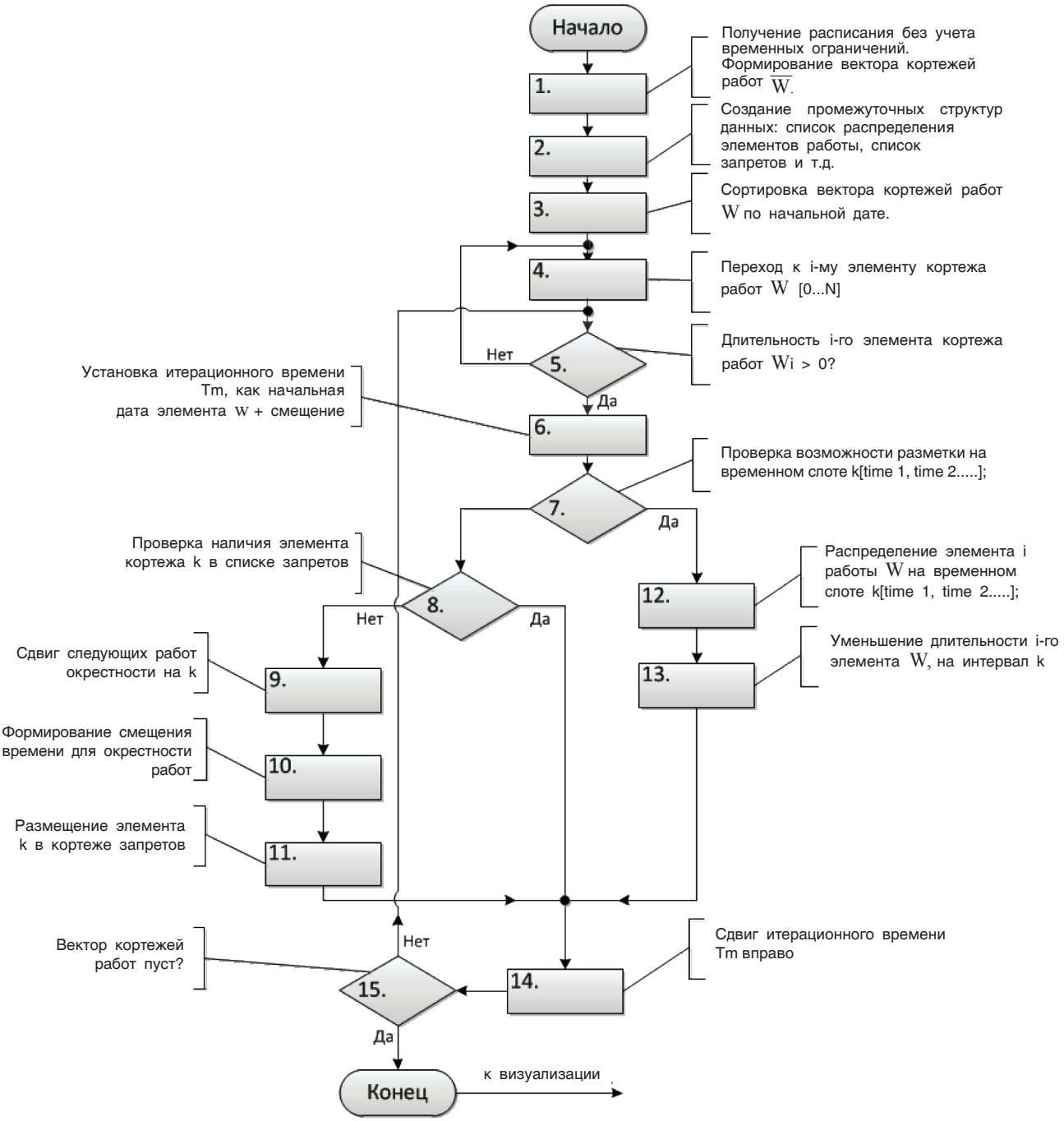

Рис. 3. *Блок-схема алгоритма разметки (labeling) работ*

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.