A_j и A_k ($j,k=\overline{1,n}$) ищутся такие ребра (d,c) и (f,g), чтобы $-l_{de}-l_{fg}+l_{df}+l_{eg}$, где l_{dc} - длина ребра (d,c), принадлежащего контуру A_j, l_{fg} - длина ребра (f,g), принадлежащего контуру A_k, l_{df}, l_{cg} - длины добавляемых ребер, было минимальным. Получившее значение составляет оценку C_{jk}, которая представляет собой критерий для выбора склеиваемых контуров A_j и A_k.

Далее, при помощи просмотра всех получившихся оценок, выбирается минимальная оценка C_{jk}, которая говорит о том, что в результате склейки контуров A_j и A_k получится контур минимальной длины из всех возможных.

Таким образом, для того, чтобы решить задачу коммивояжера для n точек, потребуется $n-1$ операций склейки (объединения) контуров.

Сразу же можно предложить существенное улучшение этого алгоритма. На каждом шаге кристаллизации не гарантировано, что контур, полученный в результате склейки, будет оптимальным. Поэтому после каждой склейки для получившегося контура можно выполнять процедуру k - оптимизации. Это улучшит качественные показатели решения, но ухудшит временные характеристики алгоритма.

Главным преимуществом алгоритма кристаллизации является то, что он предназначен для решения как симметричной задачи коммивояжера, так и несимметричной.

Для того, чтобы сравнивать различные методы решения некоторой задачи, необходимо провести как теоретическое сравнение, так и практическое сравнение, т. е. пользуясь данными, полученными в результате эксперимента.

Теоретическое сравнение по объему используемой памяти и трудоемкости для некоторых алгоритмов решения задачи коммивояжера приведено в таблице 1.

Из таблицы видно, что самыми трудоемкими методами являются точные методы – метод полного перебора и ветвей и границ. Причем, метод ветвей и границ менее трудоемок за счет того, что количество используемой им памяти гораздо больше, нежели у метода полного перебора.

Приближенные методы решения имеют примерно одинаковые трудоемкость и количество используемой памяти. Разница заключается лишь в оценке качества решения, которую можно получить лишь экспериментальным методом.

Экспериментальный анализ методов будем проводить по следующим критериям:

1). Средний объем используемой памяти.
2). Среднее время работы алгоритма.
3). Среднее отклонение длины пути от эталона. В качестве эталона выберем результат, получившийся в результате работы алгоритма кристаллизации. Отклонение будем выражать в процентном отношении, т.е. получившееся положительное значение x будет означать, что длина пути у исследуемого метода больше на $x\%$, чем у пути, получившего в результате применения алгоритма кристаллизации.

В результате эксперимента для количества точек 5, 7, 10, 15, 20, 50, 100, 250, 500, 750, 1000, получены результаты, показанные в таблицах 2 – 4. Указанное время – в секундах, объем памяти – в байтах, отклонение – в процентах.

Из приведенных выше таблиц можно сделать вывод о применимости того или иного метода. Если нужно точное решение, необходимо применять либо метод ветвей и границ, либо метод полного перебора. Метод полного перебора самый медленный, однако, в отличие от метода ветвей и границ, использует мало памяти. Эксперимент показал, что максимальное количество точек, для которого можно решить задачу коммивояжера за приемлемое время для метода перебора – 12, для метода ветвей и границ – 20.

Если точность не играет большой роли или количество точек больше 20, необходимо использовать приближенные методы решения. Так как приближенные методы решения используют примерно одинаковый объем памяти, есть смысл сравнивать их только по быстродействию и по качеству решения. Из таблицы 3 видно, что самым быстрым является метод включения ближайшего соседа. Самым медленным – приближенный алгоритм ветвей и границ. Однако, метод кристаллизации – самый точный.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Меламед И. И., Сергеев С. И., Сигал И. Х., Задача коммивояжера. Вопросы теории // Автоматика и телемеханика. – М.: Наука, 1989. №9. с.3-33.
2. Гимади Э. Х., Глебов Н. И., Перепелица В. А., Алгоритмы с оценками для задач дискретной оптимизации, Сб. «Управляемые системы», Новосибирск, вып. 11, 1974, 35-41.
3. Меламед И. И., Сергеев С. И., Сигал И. Х., Задача коммивояжера. Точные методы // Автоматика и телемеханика. – М.: Наука, 1989. №10. с.3-29.
4. Меламед И. И., Сергеев С. И., Сигал И. Х., Задача коммивояжера. Приближенные алгоритмы // Автоматика и телемеханика. – М.: Наука, 1989. №11. с.3-26.

УДК 681.51

Ревотюк М.П., Тихомирова Е.В.

АЛГОРИТМИЧЕСКАЯ ИНТЕРПРЕТАЦИЯ ПРОЦЕССОВ НА ВРЕМЕННЫХ СЕТЯХ ПЕТРИ

Известно, что наиболее привлекательным видом моделирования дискретных процессов с регулярной структурой являются сети Петри и их расширения [1]. Ориентация сетей Петри на отражение свойства восприимчивости реальных систем к локальным изменениям переменных состояния весьма удобна как при формализации дискретных процессов со сложными асинхронными взаимодействиями, так и реализации технологий объектно-ориентированного проектирования и программирования [2]. Цель исследования здесь - построение эффективных алгоритмов интерпретации процессов

Ревотюк Михаил Павлович. *Доцент каф. ИТАС Белорусского государственного университета информатики и радиоэлектроники.*

Тихомирова Екатерина Валерьевна. *Аспирант Белорусского государственного университета информатики и радиоэлектроники.*

Беларусь, г. Минск.

на сетях с учетом вычислительной эффективности по критериям памяти и быстродействия. Необходимость уделения пристального внимания вопросам вычислительной эффективности возникает, например, при конструировании моделей замещения объектов управления или при реализации контура управления в системах организационно-технологического уровня. Особенность использования сетевых моделей - необходимость учета динамики программного отображения траектории процессов на сети в реальном времени. Здесь приходится связывать модельные и реальные процессы с учетом динамических характеристик вычислительной среды.

1. ФОРМАЛЬНОЕ ОПРЕДЕЛЕНИЕ ВРЕМЕННЫХ СЕТЕЙ ПЕТРИ

Под временной сетью Петри с задержками в переходах ***(TPN)*** будем понимать шестерку :

$$TPN=(A, B, D, V, W, C),$$

где A и B - это конечные множества элементов, называемые переходами и позициями, причем $A \cap B=0$, а $A \cup B \neq 0$,

D – конечное множество описаний задержек переходов, определенная на множестве неотрицательных рациональных чисел $R+ : D=A \rightarrow R+$,

V – это функция инцидентности, отображающая связь между позициями и переходами: $V=(A*B) \cup (B*A) \rightarrow \{0, 1\}$,

W – это весовая функция, определенная на множестве положительных чисел $N+: W=V \rightarrow N+$,

C – это начальная разметка позиций сети, определенная на множестве неотрицательных чисел $N: C_a: B \rightarrow N$.

Функционирование сети Петри любого вида заключается в переходе от разметки C_k к разметке C_{k+1}, вследствие срабатывания переходов: $C_k=\{C_k(b), b \in B\}, k>0$. Здесь индекс k в обозначении вектора разметки означает номер этапа функционирования ***TPN***, связанного с моментом свершения особого события.

На траектории процесса функционирования ***TPN*** можно выделить особые события двух видов:

активизация перехода;

выход перехода из активного состояния или его пассивизация через интервал $d(a) \in D, a \in A$.

Обозначим для каждого элемента ***TPN*** $x \in A \cap B$ множество входных элементов $\bullet X=\{y/y \cup x\}$ и множество выходных элементов $X\bullet=\{y/x \cup y\}$.

Интерпретация процесса функционирования сети Петри в общем случае включает определение:

условий активизации переходов;

действий, реализуемых при активизации и пассивизации переходов.

Условия активизации переходов в ***TPN*** при некоторой разметке C_k: переход $a \in A$ будет активизирован тогда и только тогда, когда $C_k(b) \geq w(b, a)$ для $\forall b \in \bullet a$, где $w(b, a)$ – это вес связи позиции $b \in B$ и перехода $a \in A$.

Активизация перехода, а в момент времени $S_k(a)$ на этапе k приведет к новой разметке C_{k+1}, определяемой выражением:

$$C_{k+1}(b)=C_k(b)-w(b, a) \text{ для } \forall b \in \bullet a,$$
$$C_{k+1}(b)=C_k(b) \text{ для } \forall b \in B/\bullet a.$$

Активизированный переход а после задержки во времени на интервал $d(a) \in D$ в момент времени $f_e(a)=S_k(a)+d(a)$ на этапе $e>k$ перейдет в пассивное состояние. Если в момент времени $f_e(a)-0$, предшествующей пассивизации перехода a разметка сети была C_e, то пассивизация перехода приведет к новой разметке C_{e+1}, определяемой выражением:

$$C_{e+1}(b)=C_e(b)+w(a, b) \text{ для } \forall b \in a\bullet,$$
$$C_{e+1}(b)=C_e(b) \text{ для } \forall b \in B/a\bullet,$$

где $w(a, b)$ W – вес связи перехода $a \in A$ и $b \in B$.

Если положить $d(a)=0$ для любых $a \in A$, то получим формальное определение классической сети Петри.

Очевидно, что переход от этапа к этапу процесса функционирования ***TPN*** определяется только локальными связями переходов. Учитывая, что значение $s.(a)$ и $f.(a)$ связаны известным значением $d(a)$ для любых $a \in A$. Алгоритм интерпретации процессов на ***TPN*** можно представить по рекуррентной схеме регулярного применения правил срабатывания и последующей пассивизации переходов. Состояние процесса при этом представляется списком событий $\{s.(a),\ldots,f.(a)\}$, представляемым активизируемые переходы и моменты их пассивизации.

Для удобства рекуррентной интерпретации процессов сети введем понятия:

а) стартовый переход – дополнительный переход s, где:

$$\bullet a_s=0, a_s\bullet=\{ b/Ca(b)>0, b \in B \},$$
$$w(a_s, x)=Ca(b), x \in \bullet a_s, d(a_s)=0,$$

б) финишный переход – дополнительный переход f, где:

$$a_f=\{ b/Cn(b)>0, b \in B \}, a_f\bullet=0,$$
$$w(x, a_f)=Cn(b), x \in a_f\bullet, d(a_f)=0.$$

Таким образом, сеть, дополненная переходами s и f, имеет нулевую начальную и конечную разметки и далее будет называться стандартной.

Цель введения понятия стандартного вида ***TPN*** - рекуррентная интерпретация процесса на сети относительно перехода a_s. В этом случае описание сети может быть загружено непосредственно перед активизацией a_s и выгружено в момент пассивизации a_f. Это открывает возможность введения понятия вложенных и прерываемых сетей. Практика моделирования реальных процессов требует обсуждения вопросов использования такого рода сетей [3].

2. АЛГОРИТМИЧЕСКАЯ ИНТЕРПРЕТАЦИЯ TPN

Создание интерпретатора процесса на ***TPN*** порождает следующие задачи :

представление исходного описания ***TPN***;

разработка структуры данных статического описания ***TPN***;

разработка алгоритма динамического описания процесса на ***TPN***;

Пример решения задачи представления исходного описания:

пусть ***TPN*** в стандартной форме имеет вид (рисунок 1):

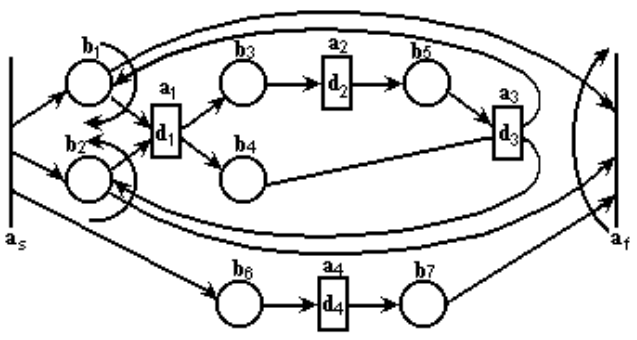

Рисунок 1 – Пример ***TPN*** стандартного вида.

Для ввода в ЭВМ подходящей формой описания ***TPN*** является ее представление как двудольного нагруженного ориентированного графа с раздельным описанием структуры сети и переходом [2].

Описание графа удобно представить в виде структуры смежности ***S0***. Описание структуры ***S0*** включает в себя описание связей позиций и связей переходов ***TPN***:

//описание связей переходов:

a_s: b_1 ; b_2 ; $b6$

a_1: b_3 ; b_4

a_2: b_5

a_3: b_2 ; b_1

a_4: b_7

//описание связей позиций

b_1: a_f ; a_1

b_2: a_f ; a_1

b_3 : a_2

b_4 : a_3

b_5 : a_3

b_6 : a_4

b_7 : a_f

Используется следующая схема кодирования описания узла графа :

$$x_i\text{: } [y_1, w_1\ ]\ ;[y_2, w_2\ ]; \ldots\ [y_n, w_n\ ].$$

Для двудольного графа после фиксации типа одной из вершин, например a_s, легко определить типы остальных вершин. В результате сканирования области ***S0*** легко определить множество ***A***, ***B***, а также функции ***W, V***. Описание перехода можно указать представлением оператора вызова функции вычислением задержки перехода, например в виде

$$<x\text{: } k, P1_k, P2_k, \ldots>,$$

где ***x*** - переход ***TPN***, ***k*** - идентификатор функции, Pi_k - значение фактического параметра.

Перейдем к преобразованию исходного описания в структуры статического описания ***TPN*** пригодные для построения процесса во времени.

По определению ***TPN*** все события в процессе ее функционирования однородны.

Пусть в некоторый момент времени $t=min\{t_i \in L_k\}$ зафиксирован факт пассивизации некоторого перехода, что можно выявить на основе списка событий

$$L_k=\{< t_i, a_i >, i > k\text{-}1\},$$

где t_i - момент пассивизации перехода a_i. Обработка последствий пассивизаций перехода требует наличие следующих процедур:

D1 – моделирование действий перехода ***a*** при передаче меток в позиции $b \in a\bullet$ (передачу меток необходимо выполнять до рассмотрения других фаз процесса на ***TPN*** для отображения фактора одновременности для изменения разметки в выходных позициях перехода);

D2 – организация обработки последовательностей изменения разметки в позиции $b \in a\bullet$ (последовательность просмотра таких позиций определяется ранее рассмотренными способами устранения конфликтов);

D3 – проверка возможности активизации новых операторов из множества: $\{x \in \bullet y, y \in b, b \in a\bullet\}$;

D4 – активизация перехода ***x*** если необходимость этого установлена процедурой ***D3***. Активизация перехода x требует изменения разметки в позициях $\bullet x$ и планирование нового события в момент $t+d(x)$;

D5 – фиксация особого события $<t+d(x),\ x\ >$ в списке событий;

D6 – выбор основного события на ***TPN*** и начало нового цикла моделирования если список событий не пуст.

Алгоритм моделирования процессов на ***TPN*** в результате представляется в следующем рекуррентном виде:

$$L0 = <0, a_s>, a_s \in A\ ;$$

$$L_{k+1}=D_6(D_1, D_2\ (D_3\ (\ D_4\ (\ D_5\ (L_k))))),\ k>0.$$

Время между событиями может быть привязано к реальному, если организовать ожидание между моментами пассивизации переходов.

Перейдем к определению структур данных статического описания ***TPN***.

Описание разметки позиций представимо структурой данных $S1=\{c_i,\ i \in B\}$ – линейным одномерным массивом, компактность представления которого зависит от способа идентификации элементов сети.

Описание переходов и их временных задержек представлено структурой $S2=\{d_i,\ i \in A\}$- неоднородным в общем случае массивом. Идентификация объектов ***TPN*** и адресация к значениям их атрибутов выполняются индексированием.

Список событий представим структурой данных $S3=\{HEAD, N, F\}$. Здесь ***HEAD*** - указатель головного элемента списка, соответствующего переходу с ближайшим моментом пассивизации относительно текущего времени;

N - комплект указателей последующих элементов списка, $N:=\{N_i,\ i \in A\}$;

F - комплект значений моментов пассивизации переходов ***TPN***, $F:=\{F_i,\ i \in A\}$.

Способ представления описания связей между вершинами ***TPN*** можно зафиксировать, выделив следующие структуры данных:

$S4=\{Iao,<Jao,Wao>\}$ - описание выходных связей переходов для моделирования последствий пассивизации переходов,

где ***Iao*** - комплект указателей списков смежности выходных связей переходов,

Jao - комплект списков выходных позиций переходов,

Wao - комплект весов выходных связей переходов.

$S5=\{Ibo,\ Jbo\}$ - описание выходных связей позиций, используемое при установлении множества переходов, потенциально активизируемых при дополнении разметки позиций,

где ***Ibo*** - комплект указателей списков смежности выходных связей позиций,

Jbo - комплект списков выходных переходов позиций.

$S6=\{Iai,\ <Jai,Wai>\}$ - описание входных связей переходов, используемое при проверке условий активизации и изменении разметки позиций после активизации переходов,

где ***Iai*** - комплект указателей списков смежности входных связей переходов,

Jai - комплект списков входных позиций переходов,

Wai - комплект весов входных связей переходов.

Оценка размера памяти для хранения статического описания временной сети Петри:

$|S1| = |B|$;

$|S2| = |A|$;

$|S3| = |N| + |F| = 2|A|$;

$|S4| = |Iao| + |Jao| + |Wao| = 1 + |A| + 2|Jao|$;

$|S5| = |Ibo| + |Jbo| = 1 + |B| + |Jbo|$;

$|S6| = |Iai| + |Jai| + |Wai| = 1 + |A| + 2|Jai|$.

Очевидно, что $|Jbo|=|Jai|$, так как каждая выходная дуга позиции является одновременно входной дугой одного из переходов сети.

Таким образом, общий объем памяти S для размещения статического описания *TPN* после суммирования :

$$S = 5|A| + 2(|B| + |Jao|) + 3(1+|Jai|).$$

Нетрудно убедиться, что, обсуждаемый вариант статического описания также явно предпочтительнее матричных представлений сетевых моделей, размерность которых $(|A|+|B|)^2$.

Описание динамических процессов на *TPN* можно представить в объектно-ориентированном стиле:

```
#include <stdio.h>
//Класс временных сетей Петри с задержками в
переходах
class TPN {
                        //Структура S1
 int   *C;             // Вектор разметки пози-
ций
                        //Структура S2
 int   *D;             // Вектор задержек пере-
ходов
                        //Структура S4
 int  *Iao,            // Указатели списков вы-
ходных дуг
      *Jao,            // Номера выходных пози-
ций
      *Wao,            // Веса выходных дуг
переходов
                        //Структура S5
 int  *Ibo,            // Указатели списков вы-
ходных дуг
     *Jbo;            // Номера выходных пози-
ций
                        //Структура S6
 int  *Iai,            // Указатели списков вы-
ходных дуг
      *Jai,            // Номера выходных пози-
ций
      *Wai;            // Веса выходных дуг
переходов
//Функции-элементы описания динамических
процессов
 void D1(void);
 void D2(int);
 void D3(int);
 void D4(int);
 void D5(int);
 void D6(int);
 protected:
 void D2(int);
                         //Структура S3
 int *N,                //Поля ссылок списка
событий
     *F;                //Моменты пассивизации
переходов
 public:
 TPN();                 //Конструктор тестовой
TPN
 void interp(){D1();}//Интерпретатор процес-
сов на TPN
 ~TPN();                //Деструктор TPN
};
#define HEAD (*N)    //Указатель списка со-
бытий
#define TEMP (*F)    //Момент свершения осо-
бого события
//Управление последовательностью внутренних
событий
void TPN::D1(void) {
//Инициализация списка событий
      HEAD=0;        //Номер стартового пе-
рехода
      TEMP=0;        //Начальный момент мо-
дельного времени
//Обработка списка событий
 do{
//Пауза в реальном времени на интервал
F[HEAD]-TEMP
  printf("\n  d=%d",F[HEAD]-TEMP);
//Выборка очередного пассивизируемого пере-
хода
  int k=HEAD;
//Коррекция списка событий
  HEAD=N[k], TEMP=F[k];
  printf("\n- k=%d, t=%d",k,TEMP);
//Пассивизация перехода
  D2(k);
 } while (HEAD>0);
}
//Обработка последствий пассивизации перехо-
да
void TPN::D2(int k){
//Изменение разметки выходных позиций пере-
хода k
for (int i=Iao[k], j=Iao[k+1]; i<j;
i++)C[Jao[i]]+=Wao[i];
//Организация попыток активизации переходов
 for (i=Iao[k]; i<j; i++)
      D3(Jao[i]);
}
//Обработка последствий изменения разметки
позиций
void TPN::D3(int k){
//Просмотр списка выходных переходов позиции
k
for (int i=Ibo[k], j=Ibo[k+1]; C[k]&&(i<j);
i++)
     D4(Jbo[i]);
}
//Попытка активизации нового перехода
void TPN::D4(int k){
//Проверка условий активизации перехода k
for (int i=Iai[k], j=Iai[k+1]; i<j; i++)
     if (C[Jai[i]]<Wai[i]) return;
//Организация активизации перехода k
     D5(k);
}
//Планирование пассивизации перехода
void TPN::D5(int k){
//Изменение разметки входных позиций перехо-
да k
for (int i=Iai[k],j=Iai[k+1]; i<j; i++)
C[Jai[i]]-=Wai[i];
//Фиксация момента пассивизации перехода k
     D6(k);
}
//Расширение списка событий
void TPN::D6(int k){
//Фиксация момента пассивизации перехода k
 int f=F[k]=TEMP+D[k];
```

```
 printf("\n+ k=%d, t=%d",k,f);
//Поиск позиции в списке событий
for (int i=0, j=HEAD; (j>0)&&(f>=F[j]); i=j,
j=N[i]);
//Коррекция связей списка событий
 N[k]=j, N[i]=k;
}
//Конструктор TPN по умолчанию
TPN::TPN(){
//Описание примера TPN
 enum places {b1,b2,b3,b4,b5,b6,b7};
 enum transition {as,a1,a2,a3,a4,af};
 enum weight {w0,w1};
 static int
 Cx[b7+1]={0,0,0,0,0,0,0},
Dx[af+1]={0,2,3,4,10,0},
 Jaox[]={ b1,b2,b6, b3,b4, b5, b1,b2, b7 },
 Waox[]={ w1,w1,w1, w1,w1, w1, w1,w1, w1 },
 Iaox[]={
0,3,5,6,8,9,sizeof(Jaox)/sizeof(*Jaox)},
 Jbox[]={af,a1, af,a1, a2, a3, a3, a4, af},

Ibox[]={0,2,4,5,6,7,8,sizeof(Jbox)/sizeof(*J
box)},
 Jaix[]={ b1,b2, b3, b4,b5, b6, b1,b2,b7},
 Waix[]={ w1,w1, w1, w1,w1, w1, w1,w1,w1},
 Iaix[]={
0,0,2,3,5,6,sizeof(Jaix)/sizeof(*Jaix)};
 C=new int[b7+1];
 for (int i=0; i<=b7; C[i++]=0);
 D=Dx;
 N=new int[sizeof(Dx)/sizeof(*Dx)];
 F=new int[sizeof(Dx)/sizeof(*Dx)];
 Iao=Iaox;
 Jao=Jaox;
 Wao=Waox;
 Ibo=Ibox;
 Jbo=Jbox;
 Iai=Iaix;
 Jai=Jaix;
 Wai=Waix;
}
//Деструктор TPN
TPN::~TPN() {
 delete N;
 delete F;
 delete C;
}
// Тестовая программа
void main() {
 TPN x;
 x.interp();
}
```

Результаты работы тестовой программы:

```
 d=0
-k=0, t=0
+k=1, t=2
+k=4, t=10
 d=2
-k=1, t=2
+k=2, t=5
 d=3
-k=2, t=5
+k=3, t=9
 d=4
-k=3, t=9
+k=1, t=11
 d=1
-k=4, t=10
 d=1
-k=1, t=11
+k=2, t=14
 d=3
-k=2, t=14
+k=3, t=18
 d=4
-k=3, t=18
+k=5, t=18
 d=0
-k=5, t=18
```

Очевидно, что вычислительная сложность процесса обработки последствий свершения отдельного события на ***TPN***: $R=O(|a\bullet||\bullet a||b\bullet|)$, $a \in A$, $b \in B$.

ЗАКЛЮЧЕНИЕ

Преимущества и полезность формальной спецификации дискретных процессов сетями Петри и их расширениями реализуются в полной мере посредством решения задачи алгоритмической интерпретации сетевых описаний. Конструктивный путь ее решения - построение специализированной имитационной системы с рекуррентной схемой алгоритма интерпретации.

Предложенные в настоящей работе схемы реализации процедур моделирования процессов на сетевых моделях обладают потенциально возможными оценками эффективности качества по критерию "память - быстродействие".

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Бусленко Н.П., Калашников В.В. Лекции по теории сложных систем. -М.: Советское радио, 1973. - 440с.
2. Питерсон Дж. Теория сетей Петри и моделирование систем. -М.: Мир, 1984, - 264с.
3. Ревотюк М.П., Иванчиков А.А. Динамическое описание процессов на сетевых моделях / БГУИР - Минск, 1996. - 20 с. - Деп. в ВИНИТИ 03.10.96, N6137.

УДК 681.51

Тихомирова Е.В.

ПОСТРОЕНИЕ МОДЕЛИ УПРАВЛЕНИЯ ДИСКРЕТНЫМИ ПРОИЗВОДСТВЕННЫМИ ПРОЦЕССАМИ

Рассматриваемые системы компаундирования предназначены для получения выходного продукта посредством смешения в заданном соотношении некоторых исходных продуктов, например на нефтехимических технологических комплексах. Все задания на смешение в установках компаундирования, формирующие портфель заказов, характеризуются типом, объемом выходного продукта и конечными временными сроками его изготовления. Маршрут выполнения задания представляет собой набор активных элементов оборудования: задвижки, смесители, насосы и т.п., по которому происходит согласованное протекание жидкостей и определяется на основе сведений о необходимом объеме требуемого вы-