МИГРАЦИЯ ПРОЦЕССА РЕШЕНИЯ ЗАДАЧ МЕТОДОМ ДИНАМИЧЕСКОГО ПРОГРАММИРОВАНИЯ

Зобов В.В., Кароли М.К.
Кафедра информационных технологий автоматизированных систем
Научный руководитель: Ревотюк М.П., к.т.н., доцент
e-mail: rmp@bsuir.by

Аннотация **—Рассмотрена задача безопасного прерывания процесса решения задачи с сохранением его состояния для продолжения процесса на другом узле гетерогенной вычислительной сети другим агентом системы.Предложен алгоритм выделения пространства состояния для задачи коммивояжера, решаемой методом динамического программирования.**

Ключевые слова: задача о назначении,устойчивость решения, метод потенциалов

Процедуры реализации метода динамического программирования, базирующиеся на использовании принципа последовательной декомпозиции задачи, пригодны для естественного распараллеливания на вычислительных сетях. Управление потоками задач при нерегламентированном режиме доступности рабочих станций сети общего назначения порождают необходимость решения проблемы грануляции и синхронизации подзадач [1].

Предмет рассмотрения – способ компактного представления в произвольный момент состояния задачи,решаемой методом динамического программирования, для последующего восстановления состояния и продолжения процесса решения на любом доступном узле сети. Идея использования системных средств не будет рассматриваться как не приемлемая в гетерогенных вычислительных средах.

Какпример, рассмотрим задачу коммивояжера с матрицей $C(i,j),\ i,j=\overline{1,n}$ [2]. Цель ее решения – поиск гамильтонова цикла минимальной длины.

Обозначим множество $J_k=\{j_m,\ m=\overline{1,k}\}$, тогда рекуррентно определяемая связь подзадач при условии размещения корня дерева подзадач в вершине 1 имеет вид

$$T(i,J_k)=\min_m\{C(i,j_m)+T(j_m,J_k\setminus j_m,\ m=\overline{1,k}\},$$

$$T(i,j)=C(i,j)+C(j,1) \qquad (1)$$

Рекурсия обхода дерева подзадач преследует цель поиска перестановки $\{1,\ j_2,\ j_3,\ldots,j_n\}$,соответствующей $T(1,\ J_{n-1}),\ n>2$.

Набор переменных состояния процесса ветвления определяется левой частью выражения (1). Нетрудно заметить, что ветвление на любом уровне возможно с сохранением порядка следования элементов множеств $J_k, k=\overline{n-1,2}$. Глубина ветвления не превосходит значения n, поэтому активные ветви дерева порождаемы из вектора $J_n=\{\overline{1,n}\}$. Как показано далее в классе tspd, что для возобновления поиска решения после прерывания требуется память объемом $O(3n)$, включающая вектор перестановки лучшего гамильтонова цикла (tspd::r), вектор представления вершин пути от корня дерева до листьев (tspd::z) и вектор позиций ветвей дерева (tspd::a):

```
template<class T>class tspd {
 void swap(int&x,int&y){ int
t=x;x=y;y=t; }
protected:
 T **c, w;
int n, *a,*r,*z;
 virtual void f(int *x, int k, int l, T
d) {
  T *e=c[z[l++]=x[0]];
if (k>2) {
int *y=x+1, m=k-1, i,t;
for (i=0; i<m; i++) {
    swap(y[0],y[i]);
    f(y,m,l,d+e[y[0]]);
   }
   t=y[0], --m;
for (i=0; i<m; y[i]=y[i+1], i++);
   y[i]=t;
  } else {
   d+=e[z[l]=x[1]]+c[x[1]][0];
if (w>d) { w=d;
for (int i=0; i<=n; i++) r[i]=z[i];
 }}}
public:
 tspd(int N,T **C): n(N),c(C) {
  if (a=newint[(++N)*3]) { r=a+N; z=r+N;
}
 }
 ~tspd() { delete[] a; }
 virtual void f() { //
Запускпроцсссапоиска
  z[0]=z[n]=0; w=W;
for (int i=0; i<n; i++) a[i]=i;
int *y=a+1, m=n-1;
  T *e=c[a[0]];
for (int i=1; i<m; i++) {
   swap(y[0],y[i]);
   f(y,m,1,e[y[0]]);
}}};
```

Здесь любая ветвь дерева проецируются на вектор tspd::a, а параметры функции .tspd::f(int,int,int,T) представляют состояние процесса поиска в момент прерывания. В производном классе отражение возможности миграции процесса производится так

```
 virtual void f(int *x, int k, int l, T
d) {
if (interrupted) {
  // Сохранение a[n],r[n],z[l],a-x,k,l,d
  exit(1);
 }
 tspd::f(x,k,l,d);
}
virtual void f() { //
Запускпроцсссапоиска
// Восстановлениеа[n],r[n],z[l],a-
x,k,l,d
f(y,m,1,e[y[0]]);
}
```

[1] Воеводин, В.В. Решение больших задач в распределенных вычислительных средах/В.В. Воеводин – Автоматика и телемеханика, 2007, № 5. – С.32-45.

[2] Gutin, G. The Travelling Salesman Problem and Its Variations/Gutin G., Punnen A. P. – Dordrecht: Kluwer Academic Publishers, 2007. – 830 p.