第20卷 第5期
1994 年 9 月

自 动 化 学 报
ACTA AUTOMATICA SINICA

Vol. 20, No. 5
Sep., 1994

多目标动态规划时段轮换并行算法

康 一 梅
（中科院自动化所 北京 100080）

吴 沧 浦
（北京理工大学自控系 北京 100081）

摘 要

针对 SIMD 和 MIMD 结构的并行机提出多目标动态规划时段轮换并行算法，多目标动态规划的时段轮换迭代算法，将全过程优化问题转化成子过程优化问题，然后在子过程非劣解集中寻找全过程非劣解. 这样，将多目标动态规划内存不足的问题转化成时间问题，然后利用并行机超高速运算的优势来有效地解决内存不足问题. 通过时间复杂性、加速比分析及实例说明了算法的有效性及优越性.

关键词：并行算法，时间复杂性，加速比，时段轮换.

1 引言

并行计算机以其超高速运算的能力越来越受到人们的重视，但要真正有效地利用并行机的优势，并行算法的研究是非常重要的. 本文分别针对 SIMD 和 MIMD 结构的并行机提出多目标动态规划时段轮换同步并行算法和异步并行算法. 并行计算机的主要优点在于提高运算速度，而动态规划的主要难题是“维数灾”带来的内存不足. 如何利用并行机来解决动态规划的内存不足？动态规划的时段轮换算法，将内存问题转化成时间问题，这样就可以利用并行机的优点来有效地解决多目标动态规划内存严重不足的问题.

2 多目标动态规划时段轮换迭代算法

一般多目标动态规划可描述成

$$\begin{aligned} &\mathrm{Cop}F(x_0,\ \delta_0),\\ &x_{k+1}=\Gamma_{k+1}(x_k,u_k),k=0,1,\cdots,N-1, \end{aligned}\tag{2.1}$$

其中N是时段数，$F\in R^p$，$\delta_0=(u_0,\ u_1,\ \cdots,\ u_{N-1})$，状态变量 $x_k\in X_k\subset R^n$，控制变量 $u_k\in U_k\subset R^m$，且满足 $x_0\in X_0\subset R^n$，$x_N\in X_N\subset R^n$.

对上述模型，常规解法的方程为

本文于 1992 年 12 月 9 日收到

$$\underset{\delta_k\in\Delta_k}{\mathrm{Cop}}\ F_k(x_k,\delta_k)=\underset{\delta_k\in\hat{\Delta}_k}{\mathrm{Cop}}\{L_k(x_k,u_k)+\underset{\delta_{k+1}\in\Delta_{k+1}}{\mathrm{Cop}}\ F_{k+1}[\Gamma_{k+1}(x_k,u_k),\delta_{k+1}]\},\quad(2.2)$$

其中

$$\delta_k=\{u_k,u_{k+1}\cdots,u_{N-1}\},$$

$$\hat{\Delta}_k(x_k)=U\{u_k,\Delta^*_{k+1}[\Gamma_{k+1}(x_k,u_k)]\},$$

$$\Delta^*_k(x_k)=\{U\{u_k,\Delta^*_{k+1}[\Gamma_{k+1}(x_k,u_k)]\}\}^*.$$

时段轮换算法的思想是[1]:对于全过程,任选其中的两个时段,固定此两时段的起始、终止时刻的状态,其余时刻的状态也都给定,如此将全过程的最优化问题转化成两时段的过程优化问题,求解此两时段的过程优化问题后,得出该后一时段起端的相应最优状态,将此状态固定,紧接的下段状态放松,以该时段及紧接的下一时段作为新的两时段问题进行优化,如此类推,继续按时段轮换进行迭代.这样就将多时段多目标动态规划问题转化成两时段寻优的多任务问题,也将空间问题转换成时间问题,从而可以利用并行处理有效地解决问题.

定理 1. 设 $\mathbb{F}(x,z)=[F_1(x,z_1),F_2(x,z_2),\cdots,F_l(x,z_l)]$ 是 $X\times Z\to R^l$ 上的函数,$z(w)$ 是$W\to Z$的函数,$Z\subset R^l$,若 $\mathbb{F}(x,z)$ 中每一分量 F_i 都对 z_i 单调非降,$z(w)$ 在W上存在 Pareto 有效解,则

$$\underset{w\in W}{\mathrm{Cop}}\,\mathbb{F}(x,z(w))\subset\mathbb{F}(x,\underset{w\in W}{\mathrm{Cop}}\ z(w)).$$

证明. 用反证法. 记 $z(w)$ 在W上的 Pareto 有效解为 $z^*(w^*)=\underset{w\in W}{\mathrm{Cop}}z(w)$,即 $\exists z\in Z$,使 $z\leqslant z^*$,亦即

$$\begin{cases}z_1\leqslant z_1^*\\ z_2\leqslant z_2^*\quad\text{且至少存在一个 } j \text{ 使}\\ \vdots\qquad\quad z_j<z_j^*,\ 1\leqslant j\leqslant l.\\ z_l\leqslant z_l^*\end{cases}$$

记 $\mathbb{F}^*(x,z(\hat{w}))=\underset{w\in W}{\mathrm{Cop}}\mathbb{F}(x,z(w))$,即 $\exists\bar{w}\in W$,使得 $\mathbb{F}(x,z(\bar{w}))\leqslant\mathbb{F}^*(x,z(\hat{w}))$,则 $\exists\bar{w}\in W$,使 $z(\bar{w})\leqslant z(\hat{w})$. 因为若 $\exists\bar{w}\in W$ 使 $z(\bar{w})\leqslant z(\hat{w})$,由单调性则有 $\mathbb{F}(x,z(\bar{w}))\leqslant\mathbb{F}^*(x,z(\hat{w}))$,与前面矛盾. 即若 $\mathbb{F}^*(x,z(\hat{w}))=\underset{w\in W}{\mathrm{Cop}}\mathbb{F}(x,z(w))$,则 $z(\hat{w})=z^*(w^*)$. 故

$$\underset{w\in W}{\mathrm{Cop}}\mathbb{F}(x,z(w))\subset\underset{w\in W}{\mathbb{F}}(x,\mathrm{Cop}z(w))$$

得证.

多目标最优化往往没有绝对最优解,一般是求得非劣解集 (Pareto 有效解集),供决策者从中选择. 由定理 1 知全过程非劣解一定是子过程非劣解,因此时段轮换迭代算法是先求子过程非劣解,然后在子过程非劣解集中求全过程非劣解,即每次迭代都是一两层寻优过程.

2.1 内层寻优

内层寻优即求子过程非劣解集. 设 $i+1$ 次迭代是对第 k 段寻优,迭代初始集为 X^i,任取 $\xi^i\in X^i$,构造以下状态序列集

$$\xi^i_k=\{x_0,x_1,\cdots,x_{k-1},x_{k+1},\cdots,x_N\},$$

其中 x_j 是 ξ^i 中第 j 段状态,$1\leqslant j\leqslant N,j\neq k$. 记

$$X_k^i = \{\xi_k^i \mid \xi_k^i \text{ 如上定义，} \xi^i \in X^i\}.$$

本次迭代是对第 k 段寻优，x_k 在 X_k 中变动，以 X_k^i 作为初始状态序列集，内层寻优即求

$$\underset{\xi_k^i \in X_k^i}{\mathrm{Cop}} \{L_{k-1}(x_{k-1},u_{k-1}) + L_k(x_k,u_k) \mid x_k = \Gamma_k(x_{k-1},u_{k-1}),$$

$$x_{k+1} = \Gamma_{k+1}(x_k,u_k),\text{. 且 } x_{k+1} \in \xi_k^i\}. \tag{2.3}$$

由此得出相应于每一个 ξ_k^i 的第 k 段非劣解 x_k'，得到解集 $X^{i+1\prime} = \{\xi^{i+1} \mid \xi^{i+1} = \{x_0, x_1, \cdots, x_{k-1}, x_k', x_{k+1}, \cdots, x_N\}, \xi_0^i \in X^i\}$。显然全过程相应于每一个 ξ^{i+1} 的指标值仍有优劣之分，故应对所有 ξ^{i+1} 寻优。

第 k 段子过程对一个初始序列 ξ_k^i 的内层寻优算法如下，其中 n 和 n_1 是离散化格点数。

内层寻优串行算法：

1) $i = 1$.

2) $x_k' = \Gamma_k(x_{k-1}, u_{k-1}^i)$.

3) $i1 = 1$.

4) $x_{k+1}' = \Gamma_{k+1}(x_k', u_k^{i1})$.

5) 若 $x_{k+1}' \notin X_{k+1}$，转向 8).

6) $V_k = L_{k-1}(x_{k-1}, u_{k-1}^i) + L_k(x_k', u_k^{i1})$.

7) 调比较子程序，判断是否非劣解。若是，则存起来.

8) 若 $i1 = n_1$，则转向 10).

9) $i1 = i1 + 1$，转向 4).

10) 若 $i = n$，转向 12).

11) $i = i + 1$，转向 2).

12) 结束(转向下一操作).

比较子程序算法与外层寻优算法很相似，为节省篇幅，这里不作详细说明.

2.2 外层寻优

外层寻优是在子过程非劣解集中求全过程非劣解。即求

$$\underset{\xi_k^i \in X_k^i}{\mathrm{Cop}} \{U\{L_{k-1}(x_{k-1},u_{k-1}) + L_k(x_k,u_k) \mid \xi^{i+1} \in X^{i+1\prime},$$

$$\xi^{i+1} = \{x_1, x_2, \cdots, x_k', \cdots, x_N\}\}\}.$$

两层寻优可表示为

$$\underset{\xi_k^i \in X_k^i}{\mathrm{Cop}} \ \underset{x_k \in X_k'}{[\mathrm{Cop}\, F_k(x_k,\xi_k^i)]},$$

其中 X_k' 是所有 x_k' 的集合.

外层寻优的算法。$F_k^l = [F_{k_1}^l, F_{k_2}^l, \cdots, F_{k_p}^l]^{\mathrm{T}}$，$F_k^l$ 是第 l 个初始向量或比较中所得的非劣目标向量，m_k 是初始向量或已得非劣目标向量的个数，$l = 1,2,\cdots,m_k$.

外层寻优串行算法：

1) $l = 1$.

2) 令 $k1 = k2 = k3 = 0$.

3) $i = 1$.

4) 若 $F_{k_i} < F^l_{k_i}$, 转向 6).

5) 若 $F_{k_i} = F^l_{k_i}$,转向 7),否则转向 8).

6) 令 $k1 = k1 + 1$,转向 8).

7) $k2 = k2 + 1$.

8) 若 $i = p$,转向 9). 否则,令 $i = i + 1$,转 4).

9) 若 $k2 = p$,转向 13).

10) 若 $k1 + k3 \neq p$,转向 13).

11) 将 F^l_k 从初始向量中删除.

12) 令 $k3 = k3 + 1$, $m_k = m_k - 1$.

13) 若 $l = m_k$, 转向 15).

14) $l = l + 1$,转向 2).

15) 若 $k3 = 0$,转向 17).

16) $F^{m_k}_k{}^{+1} = F_k$.

17) 结束.

3 多目标动态规划时段轮换并行算法

计算机可按不同的原则分类，Flynn 按指令流、数据流将计算机分成四 类[2]，SIMD 和 MIMD 是其中的两种类型,一般并行机即指这两种结构的计算机. 下面分别讨论针对 MIMD 和 SIMD 结构的并行机的多目标动态规划时段轮换并行算法.

为方便起见,针对两端固定的多目标动态规划讨论. 设时段数为 8,指标数为 4,并行机的等效处理机个数为 4,状态维数为 4.

3.1 针对 MIMD 结构的时段轮换异步并行算法

MIMD 结构的并行机是在整体控制下由两个或多个具有独立运算功能的处 理 机 组成,各处理机在同一时刻执行不同的指令,指令来自不同的处理机本身，各处理机不一定要同步运算,因此可称作异步并行机. 复杂的程序被分成若干可独立运行的程序,在不同的处理机上分别执行,因而实现了任务、指令、数组各个级别的全面并行计算.

多目标动态规划时段轮换异步并行算法,是将每个子过程的整个迭代过程,即内层寻优和对该内层寻优所得子过程非劣解的外层寻优，当作一个任务块，采用任务级并行处理. 迭代开始时,先固定 0,2,4,6,8 段的状态,放松 1,3,5,7 段的状态，分别在处理机 1,2,3,4 中执行相应的程序,对 1,3,5,7 段寻优. 这四个时段的寻优不一定同时结束，先处理完的处理机要等待相邻时段处理完毕. 只要相邻两个时段(时段数差2)都处理完毕,就可在序号小的处理机中进行下一子过程的寻优，直到某一处理机的迭代得到全过程非劣解集,则停止所有处理机的迭代.

异步并行算法如下:

1) 同时在 4 个处理机中对 1,3,5,7 段寻优.

2）若第 i 个处理机对第 i 段寻优结束，且获得最优解，则停机。

3）第 i 个处理机发出第 i 段寻优结束信号。

4）若第 k 个处理机对第 $i-2$ 段寻优结束，则第 $\min(j,k)$ 个处理机对第 $i-1$ 段寻优。

5）若第 l 个处理机对第 $i+2$ 段寻优结束，则第 $\min(j,l)$ 个处理机对第 $i+1$ 段寻优。

6）转向 2)。

其中 $1\leqslant j,k,l\leqslant 4, i=1,2,\cdots,7$。

3.2 针对 SIMD 结构的时段轮换同步并行算法

SIMD 结构的并行机，每个处理机都同时执行相同的指令，因此要将子过程继续划分，进行任务级、数组级并行交错处理。

3.2.1 内层寻优

对(2.1)式给出的模型，子过程模型相同，因此寻优过程也相同。但每时段迭代所得非劣解个数不同，遇判断各段的结果也不同。因此，不能象异步并行算法那样直接将串行算法的迭代任务分配给各处理机执行。SIMD 并行机数组级并行性好，因而可将任务继续划分成更小的任务块，遇判断时逐段判断，进行数组级并行运算。

指令相同时分时段并行运行。由于初始迭代集元素不止一个，每段迭代所得的非劣解个数也不同，因此每个处理机迭代结束时，应判断未结束处理机进行的子过程未迭代的初始序列数是否为零，若不为零，则将未迭代的初始序列的迭代分给该空闲处理机进行。这样就不会有严重的负载不平衡。

遇判断时按时段依次进行数组级并行判断。

对一个初始序列的内层寻优同步并行算法。

1）四个处理机同时令 $i=1$。

2）同时计算 $x'_k=\Gamma_k(x_{k-1},u^i_{k-1}), k=1,3,5,7$ 或 $k=2,4,6$。

3）同时令 $k=1$。

4）同时判断是否 $x'_{kj}\in X_{kj}, j=1,2,3,4$。若是，则转向 6)

5）$XX(k)=0$，转 7)。

6）$XX(k)=1$。

7）若 $k=7$ 或 $k=6$，转向 9)。

8）$k=k+2$，转向 4)。

9）同时令 $i1=1$。

10）同时判断是否 $XX(k)=0$，是则该机停，等待下次计算。$k=1,3,5,7$ 或 $k=2,4,6$。

11）同时计算 $x_{k+1}=\Gamma_{k+1}(x'_k,u^{i1}_k), k=1,3,5,7$ 或 $k=2,4,6$。

12）同时令 $k=1$。

13）同时判断是否 $x'_{k+1,i}\in X_{k+1,i}, j=1,2,3,4$。是则转向 15)。

14）$XX(k)=0$，转向 16)。

15）$XX(k)=1$。

16) 若 $k=7$ 或 $k=6$,转向 18).

17) $k=k+2$,转向 13).

18) 同时判断 $XX(k)$ 是否为零,是则该机停,等待下次计算.

19) 同时计算 $V_k=L_{k-1}(x_{k-1},u^i_{k-1})+L_k(x'_k,u^{i1}_k)$.

20) 判断是否非劣解(同外层寻优).

21) 若 $i1=n_1$,转向 23).

22) $i1=i1+1$,转向 11).

23) 若 $i<n$,则 $i=i+1$,转向 2). 否则停机.

3.2.2 外层寻优

将现已求得的所有解对应的目标向量的各分量分别由各处理机进行比较运算，按偏序关系判断是否是劣解,若是则删去,否则存起来继续比较,直至删去所有劣解,即得全过程非劣解.

4 时间复杂性及加速比

4.1 串行计算时间复杂性

1) 外层寻优

设一个子过程对一个初始序列内层寻优所得的非劣解个数为 kl，则对这 kl 个子过程非劣解外层寻优的操作步数为

$$T_{1外}=\sum_{i=1}^{kl}[4+(7p+10)m_i]=4kl+(7p+10)\sum_{i=1}^{kl}m_i.$$

其中 m_i 是判断第 i 个解是否全过程非劣解时已有的全过程非劣解个数.

2) 内层寻优

设第 $i1-1$ 步所得非劣解个数为 k_{i1-1}，则一个子过程对一个初始序列内层寻优所需总的操作步数为

$$T_{1内}=\left[8+8n_1+(7p+10)\sum_{i1=1}^{n1}k_{i1-1}\right]n.$$

总的串行计算一个子过程对一个初始序列寻优所需的操作步数为

$$T_1=T_{1内}+T_{1外}=(8+8n_1)n+4kl+(7p+10)\left(\sum_{i=1}^{kl}m_i+n\sum_{i1=1}^{n_i}k_{i1-1}\right).$$

4.2 异步并行计算时间复杂性

设处理机的空闲时间为 $T_{闲异}$，则异步并行计算一个子过程对一个初始序列迭代所需的操作步数为

$$T_{4异}=\frac{1}{4}(T_1+T_{闲异}).$$

4.3 同步并行计算时间复杂性

外层寻优

$$T_{4同外}=kl+9.5\sum_{i=1}^{kl}m_i=\frac{1}{4}T_{1外}.$$

内层寻优。

$$T_{4\text{同内}} = \frac{1}{4}(T_{1\text{内}} + T_{\text{闲同}}),$$

则

$$T_{4\text{同}} = T_{4\text{同外}} + T_{4\text{同内}} = \frac{1}{4}(T_1 + T_{\text{闲同}})$$

4.4 加速比

由以上分析，异步并行算法的加速比为

$$S_{\text{异}} = \frac{T_1}{T_{4\text{异}}} = \frac{4T_1}{T_1 + T_{\text{闲异}}}.$$

同步并行算法的加速比为

$$S_{\text{同}} = \frac{T_1}{T_{4\text{同}}} = \frac{4T_1}{T_1 + T_{\text{闲同}}}.$$

由于同步并行计算处理机空闲一般在内层循环中，而异步并行计算处理机空闲是一次迭代结束后的等待，因此一般有 $T_{\text{闲异}} < T_{\text{闲同}}$，故一般情况下 $S_{\text{异}} > S_{\text{同}}$，但同步并行算法简单一些。若 $T_{\text{闲}} \ll T_1$，则有 $S \approx 4$，即加速比可近似达到 4 倍，这是加速比的极限，即处理机个数。

5 实例

通过一个典型的二维状态、二维控制三阶段线性双指标问题来说明并行计算的优越性。

设 $x_{k+1} = Ax_k + u_k$，且 $x_k \in [-2,2], u_k \in [-5,5], k = 0,1,2,3, x_0 = x_3 = [1,1]^T$，求

$$\min \begin{pmatrix} f_1 \\ f_2 \end{pmatrix} = \min \begin{pmatrix} \sum_{k=0}^{2} (x_k^T Q_1 x_k + u_k^T R_1 u_k) + x_3^T Q_1 x_3 \\ \sum_{k=0}^{2} (x_k^T Q_2 x_k + u_k^T R_2 u_k) + x_3^T Q_2 x_3 \end{pmatrix},$$

其中 $A = \begin{pmatrix} 3 & 1 \\ 2 & -1 \end{pmatrix}$, $Q_1 = \begin{pmatrix} 3.38 & 2.6 \\ 2.6 & 2.0 \end{pmatrix}$, $Q_2 = \begin{pmatrix} 4.9 & 1.4 \\ 1.4 & 0.4 \end{pmatrix}$, $R_1 = \begin{pmatrix} 7.4 & -3 \\ -3 & 2.1 \end{pmatrix}$, $R_2 = \begin{pmatrix} 3.3 & 1.5 \\ 1.5 & 2.0 \end{pmatrix}$.

当取状态变量和控制变量离散化步长为 0.5 时，在 286 微处理机上所需时间为 32 分钟，若用并行机则只需 8—10 分钟(处理机数为 4)。

实例表明，多目标动态规划时段轮换并行算法不仅解决了动态规划问题中的内存不足，而且可以保证速度，使以前无法解决的大规模多目标动态规划问题得到有效的解决。

参 考 文 献

[1] 吴沧浦. 最优控制的理论与方法. 北京: 国防工业出版社,1989. 145—146.
[2] 刘重庆. 并行处理机与应用. 上海交通大学出版社,1990. 14—16.

TIME CYCLING PARALLEL ALGORITHM OF MULTIOBJECTIVE DYNAMIC PROGRAMMING

KANG YIMEI
(*Institute of Automation, Chinese Academy of Sciences Beijing* 100080)

WU CANGPU
(*Department of Automatic Control, Beijing Institute of Technology* 100081)

ABSTRACT

Time cycling parallel algorithm of multiobjective dynamic programming are presented for SIMD and MIMD parallel computers, respectively. The time cycling algorithm of multiobjective dynamic programming is such a method that transforms the optimization problem of whole procedure into the optimization problem of subprocedures, then finds the non-inferior solutions of whole procedure in the set of the non-inferior solutions of subprocedures. With this idea, the problem of memories of multiobjective dynamic programming can be transformed into a problem of time, then the problem can be solved effectively through parallel computers with very high computing speeds.

Key words: Parallel algorithm; time complexity; speedup; time cycling.

康一梅　1968 年生．1988 年毕业于北方工业大学，1991 年于北京理工大学获硕士学位，现为中科院自动化所博士生．目前主要研究方向是并行算法设计，并行处理机系统任务分配等．

吴沧浦　1932 年生．1952 年毕业于清华大学，1962 年中国科学院研究生毕业．1981 年任北京理工大学教授，1984 年任自动控制理论及应用专业博士生导师．主要研究领域为系统最优化，大系统控制与决策，神经网络技术与智能控制等．