文章编号: 1001-0920(2006)11-1214-05

基于解构造图拆分的并行蚁群算法

刘 泓, 李 平, 闻 育
(浙江大学 a. 工业控制技术国家重点实验室, b. 智能交通交叉研究中心, 杭州 310027)

摘 要: 应用蚁群优化算法求解复杂大规模多阶段决策问题时, 其计算量会随着阶段数和各阶段离散化容许决策集合规模的增加成指数增长, 造成无法在单PC 中进行计算. 针对这一问题, 提出了基于解构造图拆分的并行蚁群算法. 该算法通过应用并行计算技术, 将解构造图拆分成若干块, 把每一块的计算任务放置在不同的 PC 上并行执行, 互相合作完成整个计算任务. 经实验验证, 这种算法可以快速有效地进行问题的求解.

关键词: 并行计算; 蚁群算法; 多阶段决策; 解构造图拆分
中图分类号: TP202 **文献标识码**: A

Parallel Ant Colony Algorithm Based on Construction Graph Decomposition

LIU Hong, LI Ping, WEN Yu
(a. National Laboratory of Industrial Control Technology, b. Research Center of Intelligent Transportation System, Zhejiang University, Hangzhou 310027, China. Correspondent: LIU Hong, E-mail: hliu@iipc.zju.edu.cn)

Abstract: To the problem that the computing capacity of ant colony optimization algorithm is exponentially increased when the stage number and decision variable dimension of complex multi-stage decision problem are increasing and can not be computed by a single PC, a parallel ant colony optimization algorithm based on the construction graph decomposition is presented. The proposed algorithm decomposes the construction graph into some parts and each part is placed on different PC. The whole computation task is accomplished by mutual cooperation in the PCs which join in the computation. Experiment result shows that the problem can be solved effectively and the computation efficiency is improved.

Key words: Parallel computing; Ant colony algorithm; Multi-stage decision; Construction graph decomposition

1 引 言

大量的复杂系统优化决策和最优控制问题都可以用多阶段决策问题的数学形式[1]来描述, 如计算机广域网路由控制问题、柔性制造系统生产调度问题、智能交通系统协调优化问题等等. 文献[2]指出由于这类问题自身的复杂性, 使得动态规划、遗传算法等经典求解方法难以实际应用, 并针对具有目标函数可加性和单调性的复杂多阶段动态决策问题, 在文献[3, 4]的基础上提出了多阶段动态决策的蚁群优化(ACO)算法. 文献[5, 6]又将其应用到城域交通实时信号灯控制优化中, 取得了理想的效果.

但随着阶段数和各阶段离散化容许决策集合的规模不断增加, 解构造图的规模以及蚂蚁在其中生成解的计算量呈指数增长的趋势, 造成无法使用单PC 进行计算. 显然, 以局域网方式互连的、基于消息传递[7, 8](MPI)的并行计算是解决这一问题的有效且可行的手段. 文献[9, 10]中提出了一种基于蚂蚁的并行蚁群优化(PACO)算法, 该算法首先在参与并行计算的每台PC 上建立相同的解构造图, 然后让若干只蚂蚁同时在不同的PC 上进行解的搜索, 每次搜索完毕后需要对不同PC 上的信息素进行同步更新, 通过蚂蚁的不断搜索给出最终的解. 基于蚂蚁的

收稿日期: 2005-08-31; **修回日期**: 2005-11-10.
基金项目: 教育部博士点基金项目(20020335106).
作者简介: 刘泓(1977—), 男, 杭州人, 博士, 从事智能交通仿真、并行计算等研究; 李平(1954—), 男, 南京人, 教授, 博士生导师, 从事智能交通、混杂系统、嵌入式系统等研究.

并行蚁群算法虽然可以提高运算速度, 但在每台PC上的解构造图的规模和蚂蚁在其中生成解的计算量与原有的保持一致, 仍无法解决上述问题

为解决该问题, 本文提出了一种基于解构造图拆分的PACO 算法 该算法首先将解构造图拆分成若干块, 然后让每一块的计算任务放置在不同的PC上并行执行, 以互相合作的方式完成整个计算任务. 通过层状构造图的拆分, 降低了每台PC 上层状构造图的规模和蚂蚁在其中进行解搜索的计算量; 而不同 PC 并行执行计算, 又提高了算法的计算速度 以一个多变量非线性时变的最优控制问题为例, 对该算法进行了仿真计算, 给出了相应的仿真结果 实验表明, 该算法具有较好的并行加速比, 能够快速有效地进行问题的求解

2　多阶段动态决策的蚁群优化算法

多阶段决策问题的数学定义为

$$\min \{J(N) = F[x(0), s_{0,N}]\}, \tag{1}$$

$$\text{s.t.}\quad x(t+1) = f[x(0), s_{0,t+1}], \tag{2}$$

$$s_{0,t+1} = \{u(0), u(1), \cdots, u(t)\}, \tag{3}$$

$$x(0) = x_0, \tag{4}$$

$$x(t) \in X(t) \subseteq R^n, \tag{5}$$

$$u(t) \in U(t) \subseteq R^r. \tag{6}$$

式中: $t = 0, 1, \cdots, N-1$; $J(N)$ 为目标函数值; F 为目标函数; $x(t)$ 为状态变量; $X(t)$ 为容许状态集合; $u(t)$ 为决策变量; $U(t)$ 为容许决策集合; $s_{0,N}$ 为问题的一个全过程策略, 简称策略, $s_{0,t+1}$ 为其子策略; 方程(2) 称为状态转移方程, f 为状态转移函数, $t+1$ 阶段的状态是由初始状态 x_0 和子策略 $s_{0,t+1}$ 决定的 当 F 或 f 具有强非线性或难以解析表达, X 或 U 为高维复杂拓扑空间时, 称其为复杂多阶段动态决策问题

ACO 算法是一种基于显式表达解空间的启发式搜索方法, 通过将优化问题的解分解成一系列的解构造块, 并将这些解构造块映射成解构造图中的节点, 从而可以用解构造图来描述优化问题的解空间 蚂蚁在解构造图中根据连接上存储的信息, 以随机方式逐步选择各个解构造块产生新的候选解, 然后将对解的评价结果以信息素的形式释放到连接上, 以引导其他蚂蚁的进一步搜索 所以, 在ACO 算法中, 蚂蚁是通过在解构造图上逐步选择解构造块生成解的方式实现对解空间的搜索

将一个具体的优化问题转换成式(1) ～ (6) 所描述的抽象数学模型后, 考虑到蚂蚁是通过一个多阶段决策过程来产生候选解的, 因此可将优化问题的解构造块集合按此多阶段决策过程划分成各个阶段的构造块集合. 一个阶段的构造块集合映射成一层节点, 整个解构造图由多层节点构成 假设优化问题的所有解都是由 N 个构造块组成, 对于解的构造块个数不等的优化问题, 可在 Y 中定义一个特殊的构造块 s_u (即单位构造块), 加上该构造块并不产生新的可行部分解 这样便把优化问题转换成具有 N 阶段的多阶段决策问题, 每个阶段的构造块集合为 $Y(t)$, $s_i = (s_i^0, s_i^1, \cdots, s_i^{n_i})$ 为解空间中的一个解, s_i^t 为解 s_i 中的一个构造块, 且 $s_i^t \in Y(t)$. 将各阶段 $Y(t)$ 中的元素与解构造图中的节点一一对应, 有

$$\Psi: s_i^t \in Y(t) \leftrightarrow v_i^t \in V^t. \tag{7}$$

该映射导出了如图 1 所示的层状解构造图, 称其中每一水平层的节点为决策层, 对应某阶段离散化容许决策集合. V^t 为第 t 层的节点集合, v_i^t 为该层的第 i 个节点 在该层状解构造图中, 蚂蚁从初始节点 v_0 出发, 重复地选择上一层某个节点, 最终形成一条路径 $\{v_0, v_{n_0}^0, \cdots, v_{n_t}^t, \cdots, v_{n_{N-1}}^{N-1}\}$, n_t 为第 t 阶段选择的节点序号, 该路径对应一个可行解 由于蚂蚁是逐个阶段地选择构造块以产生候选解的, 故只需定义相邻层节点之间的连接, 即 $e(v_i^{t-1}, v_i^j)$. 其他的具体描述参见文献[2, 6]. 在单 PC 机中, 上述算法是以串行的方式完成计算任务的, 在下文中又称之为串行的蚁群优化(SACO) 算法

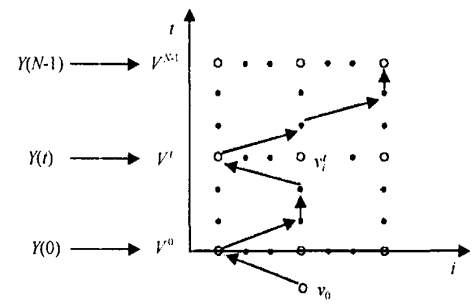


图 1　层状解构造图

3　基于解构造图拆分的 PACO 算法

文献[6] 中指出, 当各阶段离散化容许决策集合的规模为 q^r 时, SACO 算法的总计算量为 $O(CMNq^r)$. 其中: r 为每个阶段决策变量的维数, M 为蚂蚁的个数, C 为每只蚂蚁迭代的次数, N 为多阶段决策问题的阶段数 可见蚂蚁构造一个解的计算量和层状解构造图的规模是随着 r 呈指数增长, 随着 C, M, N 呈线性增长, 可见当 r, C, M, N 较大时, 在单 PC 中是无法运行该算法的, 而并行计算正是解决这一问题的有效手段

3.1　解构造图的拆分

由于 SACO 算法的解构造图是呈层状分布的, 根据这一特点, 可以对解构造图进行层状拆分. 如图 2 所示, 以图 2(a) 中实心的节点为分界点, 称这些实

心节点所在的层为分界层, 将原有的层状解构造图拆分成两个部分, 即图 2(b) 和图 2(c). 如果将图 2(b) 和 2(c) 所示的层状解构造图放置在两台 PC 上, 显然这两台 PC 可以在同一时刻运行相应的蚁群优化算法, 实现部分解的搜索. 据此, 可将 SACO 算法转变成为 PACO 算法

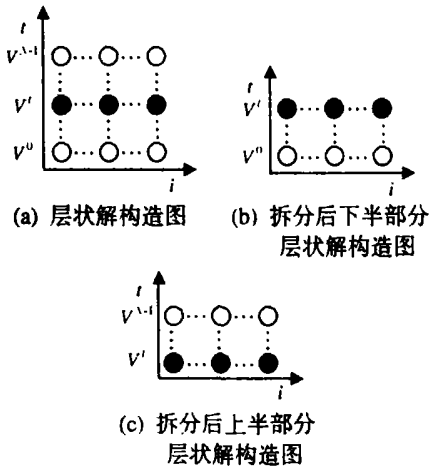


图 2 解构造图拆分示意图

在进行层状解构造图拆分时应保证各台 PC 上部分层状解构造图的规模是基本一致的, 即各台 PC 上的层数是基本相同的, 从而确保每台 PC 上的计算量是基本平衡的. 现假定参与并行计算的 PC 数为 P, 每台 PC 分别标为第 1, 2, …, P, 给出如下的层状解构造图拆分算法: 令

$$Z = \mathrm{Mod}((N + P - 1)/P),$$

If $(Z = 0)$:

$$g_i = (N + P - 1)/P,\ 1 \le i \le P,$$

If $(Z \neq 0)$:

$$g_i = \mathrm{Floor}((N + P - 1)/P) + 1,\ 1 \le i \le Z;$$

$$g_i = \mathrm{Floor}((N + P - 1)/P),\ Z < i \le P. \quad (8)$$

其中: g_i 为各 PC 上层状解构造图的层数, Mod 为取余数函数, Floor 为向 $-\infty$ 方向取整函数. 以 $N = 10$, $P = 2$ 为例, 第 1 台 PC 上的层数为 6, 第 2 台 PC 上的层数为 5, 即分界层为第 5 层, 蚂蚁在第 1 台 PC 上进行从第 0 层到第 5 层部分解的搜索, 蚂蚁在第 2 台 PC 上进行从第 5 层到第 9 层部分解的搜索.

3.2 PACO 算法

下面以两台 PC, m 只蚂蚁迭代 1 次为例, 给出

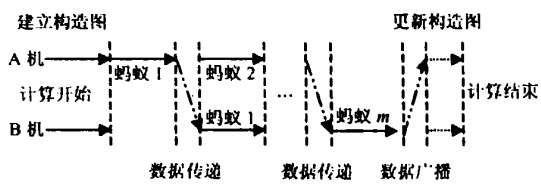


图 3 并行蚁群算法示意图

具体的 PACO 算法(如图 3 所示):

Step 1: A 和 B 两机根据层状解构造图拆分算法建立各自的层状解构造图, 分别如图 2(b) 和 2(c) 所示.

Step 2: 蚂蚁 1 在 A 机上进行搜索, 即蚂蚁 1 在第 0 层和第 t 层之间展开搜索, 此时 B 机上不进行任何计算; 当蚂蚁 1 在 A 机上搜索完毕后, 将携带在 B 机上运算所需的信息, 通过局域网传送到 B 机上.

Step 3: 蚂蚁 1 在 B 机上展开第 t 层到第 $N - 1$ 层的搜索; 此时由于 A 机空闲, 可以产生新的蚂蚁(蚂蚁 2), 让其展开第 0 层到第 t 层的搜索.

Step 4: 重复 Step 3, 直到所有的蚂蚁均搜索过一次.

Step 5: B 机上根据所有蚂蚁的搜索结果构建信息数更新数据, 并将这些数据以广播的方式传递到 A、B 两机上. 广播以后, A 和 B 两机各自根据信息数更新数据更新层状解构造图.

Step 6: A 和 B 两机更新完层状解构造图, 代表 m 只蚂蚁迭代完 1 次, 并更新了解结构图上的信息素. A 和 B 两机将停止优化算法的搜索, 在 B 机上给出最终的搜索结果, 若要在 A 机上显示, 可将上述结果传送回 A 机上.

算法的并行性主要体现在以下两点: 1) 两只蚂蚁可以在相同的时间间隔内在两台 PC 上进行搜索. 2) 两台 PC 上的层状解构造图的更新是在同一时间间隔内完成的. 可见通过采用并行计算技术, 根据解构造图拆分的原理, 将原来十分集中的计算任务和较大的内存需求分配到参与计算的各台 PC 上, 既减少了对计算能力和内存的需求, 又提高了计算速度. 同理可以将 PACO 算法由两台 PC 的情况推广到用若干台 PC 进行计算的情况. 对于 m 只蚂蚁迭代 C 次的情况, 可以通过重复 Step 1～ Step 5 C 次, 然后通过 Step 6 来结束优化算法的搜索并获得最终的结果.

4 仿真实验和分析

4.1 仿真实验

下面给出仿真实验环境: 100M 的局域网, 网络非饱和的情况下, 参与并行计算的 PC 配置一致(单 CPU, CPU 为 1.7Mhz, 内存为 512M), 编程语言为 C 语言, 数据通讯(包括数据传送和数据广播) 采用 MPI 实现. 考虑如下多输入、连续、非线性、时变动态系统的最优控制问题:

$$J(N) = \sum_{t=0}^{N-1} \{ \left| \cos[\pi x_1(t) x_2(t)/2] \right| + \sqrt{u_1^2(t) + u_2^2(t)} \}, \quad (9)$$

$$x_1(t+1) = (t+1)\sin[\pi x_1(t)/2] + x_2(t)u_1(t), \tag{10}$$

$$x_2(t+1) = (t+1)x_1(t)\cos[\pi x_2(t)/2] + u_2(t), \tag{11}$$

$|u_1(t)| \le 1;\ |u_2(t)| \le 1;\ t = 0, 1, \cdots, N-1.$

上述最优控制问题的目标函数具有可加性和马尔可夫性, 因此可以定义连接的局部启发信息为

$$\eta(v_i^{t-1}, v_i^j) = F_\eta / \{ |\cos[\pi x_1(t+1)x_2(t+1)/2]| + \sqrt{u_1^2(t) + u_2^2(t)} \},$$

其中: $\eta(v_i^{t-1}, v_i^j)$ 为连接(v_i^{t-1}, v_i^j) 的局部启发信息, F_η为调节系数, 这里取为 1; 设蚂蚁搜索到的一条路径$s = \{u(0), u(1), \cdots, u(N-1)\}$, 连接$(v_i^{t-1}, v_i^j)$ 在该路径上, 则该连接添加的信息数值为 $\Delta\tau_{ij}(k) = F_\tau / J(N)|_s$, $J(N)|_s$ 为在路径 s 下的目标函数值, F_τ 为调节系数, 此处取 10.

设$N = 10, x_1(0) = 0, x_2(0) = 0$, 决策变量采用均匀离散化方法, 并令 u_1 和 u_2 的离散阶段数为 $q = 200$(即精确到 0.01, 每层有 200^2 个节点), 参与迭代的蚂蚁个数为 20. 分别采用基于解构造图拆分的 PACO 算法($P = 2$) 和SACO 算法对问题进行求解, 各进行 100 次仿真计算, 并且每次各迭代 150 次, 其结果如表 1 所示. 实验表明, 两者的结果是接近的, 从而验证了算法的正确性. 实验中, 当 $N \ge 212$, SACO 算法在单 PC 上就无法执行, 而并行蚁群算法可以通过不断增加 PC 解决这一问题.

表 1　100 次仿真结果比较

算法类型	最好解的目标函数值 *	解的平均目标函数值
SACO	5.78	6.67
PACO	5.79	6.71

注: * 为 100 次仿真结果中最好的解

对基于解构造图拆分的 PACO 算法($P = 2$) 的具体实现给出如下几点说明:

1) SACO 算法中的一条路径 s 将被分成两个部分, 即

$$s_A = \{u(0), u(1), \cdots, u(t)\},$$

$$s_B = \{u(t), u(t+1), \cdots, u(N-1)\},$$

称 s_A 和 s_B 为部分路径或部分解.

2) 在A 机上需要记录下每只蚂蚁的编号和其搜索过的部分路径; 在B 机上不仅需要记录上述数据, 还需要记录每只蚂蚁最终获得的目标函数值.

3) 根据局部启发信息和添加的信息数的定义, A 机向B 机传送的数据包括: 蚂蚁的编号; 蚂蚁在A 机分界层上所选择的节点编号(由于分界层同时存在于A 和B 两机上, B 机可以根据节点编号, 将这只蚂蚁放置在B 机的分界层上, 从而保证蚂蚁在B 机分界层上从与A 机分界层上相同的位置展开搜索); 蚂蚁根据A 机上的部分路径计算得到的部分目标函数值 $J(N)|_{s_A}$ ($J(N)|_{s_A}$ 与B 机上部分目标函数值 $J(N)|_{s_B}$ 相加可以获得目标函数值 $J(N)|_s$).

4) B 机上广播的信息数更新数据包括: 每只蚂蚁的编号; 每只蚂蚁的目标函数值 $J(N)|_s$. 在A 和B 两机上首先根据信息数更新数据中每只蚂蚁的编号, 从记录中找出其搜索过的部分路径; 然后根据目标函数值计算出信息数添加值 $\Delta\tau_{ij}(k)$; 最后部分路径将根据这一值对信息数进行如下更新:

$$\tau_{ij}(k+1) = (1-\rho)\tau_{ij}(k) + \rho\Delta\tau_{ij}(k),$$

其中 ρ 为信息素的挥发系数, 取[0, 1] 之间的值.

4.2　仿真分析

并行加速比是评价并行算法性能的重要指标, 它的定义为

$$S_p = T_s / T_p,$$

式中: T_s 表示用 1 个处理器串行求解某个计算问题所需的时间, T_p 是用 P 个处理器并行求解该问题所需的时间. 如果并行计算中任务是平均分配的, 则根据这一定义, 可以获得基于解构造图拆分的 PACO 算法理论加速比, 即

$$\frac{MC(T_{ss} + T_{su})}{C[(P+M-1)T_{ssP} + MT_{suP} + (P+M-2)t_T + t_B]} \tag{12}$$

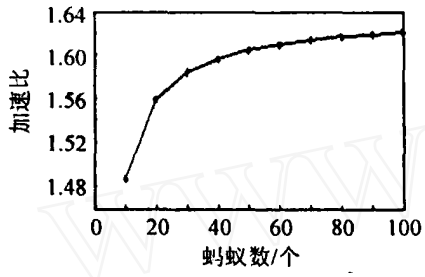
式中: T_{ss} 表示一只蚂蚁搜索一次解所需的时间; T_{su} 表示一只蚂蚁更新层状解构造图一次所需的时间; t_T 表示数据传送一次所需的时间; t_B 表示数据广播一次所需的时间; $T_{ssP} = T_{ss}/P$ 表示用 P 台 PC 进行并行计算, 每台 PC 上一只蚂蚁搜索一次部分解所需的时间; $T_{suP} = T_{su}/P$ 表示用 P 台 PC 进行并行计算, 每台 PC 上一只蚂蚁更新部分层状解构造图一次所需的时间. 从实验中, 可知本例中 $t_T \approx 2$ ms, $t_B \approx 2\times(P-1)$ ms, $T_{ss} \approx 5\times N$ ms, $T_{su} \approx 1\times N$ ms. 式(12) 可以转化为

$$\frac{1}{\frac{5}{6M} + \frac{1}{P} - \frac{5}{6PM} + \frac{2(P-1)}{3NM} + \frac{M-1}{3NM}} \tag{13}$$

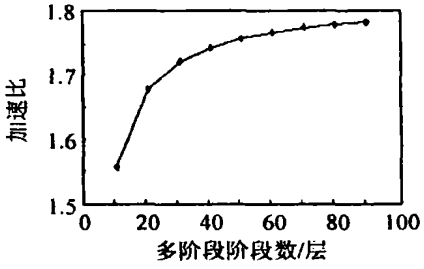
从式(13) 可知, C 与加速比无关; 如果 P 和 N 一定, M 趋向于 $+\infty$ 时, 加速比为 $3NP/(3N+P)$; 如果 P 和 M 一定, N 趋向于 $+\infty$ 时, 加速比为 $6PM/(5P+6M-5)$; 如果 N 和 M 一定, P 趋向于 $+\infty$ 时, 加速比为 0, 但在局域网内一般 PC 的数量都是有限的, 并且在算法的实际应用中, NM 一般会大于 P,

因此加速比为 0 的情况一般不会出现

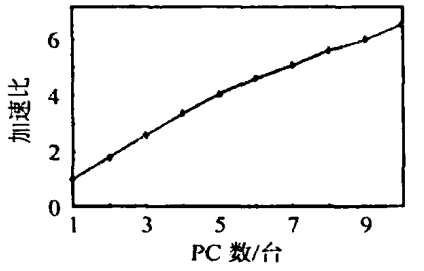
在仿真实验中, 取 $C = 150, N = 11, P = 2, M$ 分别取 10, 20, …, 100, 可得如图 4(a) 所示的实际加速比曲线; 取 $C = 150, M = 20, P = 2, N$ 分别取 11, 21, 31, …, 91, 可得如图 4(b) 所示的实际加速比曲线; 取 $C = 150, M = 20, N = 100, P$ 分别取 2, 3, 4, …, 10, 可得如图 4(c) 所示的实际加速比曲线. 从图 4 中可以看出: 1) 由各曲线的走势来看, 实验结果和理论分析基本上是一致的. 如从图 4(a) 曲线的走势可以看出, 当 N 和 P 一定时, 随着 M 的不断增大, 加速比会达到一个最大值. 2) 基于解构造图拆分的 PACO 算法适合进行较大粒度的计算(即 N, M 较大时, 不断增大 P 会提高加速比, 而当 P 一定时, 增大 N 或 M 也能提高加速比). 但是, 由于受到每台 PC 计算能力的限制, 每台 PC 有最大可计算层数的限制, 如本例中要求保证每台 PC 上的层数不超过 212.



(a) $C = 150, N = 11, P = 2, M = 10, 20, \cdots, 100$



(b) $C = 150, M = 20, P = 2, N = 11, 21, \cdots, 91$



(c) $C = 150, M = 20, N = 100, P = 2, 3, \cdots, 10$

图 4 加速比曲线

5 结 语

本文通过引入并行计算技术, 根据解构造图呈层状分布的特性, 通过对层状解构造图的拆分, 将原来十分集中的计算任务和较大的内存需求分配到参与并行计算的各台PC 上, 从而解决了多阶段动态决策的ACO 算法中随阶段数和各阶段离散化容许决策集合的规模不断增加, 解构造图的规模以及蚂蚁在其中生成解的计算量均成指数增长这一问题. 此外通过一个具有详细实验过程的仿真实例, 验证了基于解构造图拆分的并行蚁群算法的正确性. 在实验中又获得了比较理想的加速比曲线, 证明了应用该算法可以以较快的计算速度进行问题求解, 并指出了基于解构造图拆分的PACO 算法是一种适合较大粒度计算的算法.

参考文献(References)

[1] 胡运权. *运筹学教程*[M]. 北京: 清华大学出版社, 1998.
(Hu Y Q. *Tutorial of Operational Research* [M]. Beijing: Tsinghua University Press, 1998.)

[2] 闻育, 吴铁军. 求解复杂多阶段决策问题的动态窗口蚁群优化算法[J]. *自动化学报*, 2004, 30(6): 872-879.
(Wen Y, Wu T J. Dynamic-window-search ant Colony Optimization for Complex Multi-stage Decision making Problems [J]. *Acta Automatic Sinica*, 2004, 30(6): 872-879.)

[3] Dorigo M, Gambardela L M. The Ant System: Optimization by a Colony of Cooperating Agents[A]. *IEEE Trans System Man Cybernetics* [C]. 1996, B26(1): 29-41.

[4] Dorigo M, Bonabeau E, Teraulaz G. Ant Algorithms and Stignergy[J]. *Future Generation Computer Systems*, 2000, 16: 851-871.

[5] 闻育, 吴铁军. 基于蚁群算法的城域交通控制实时滚动优化[J]. *控制与决策*, 2004, 19(9): 1057-1063.
(Wen Y, Wu T J. Real-time Rolling Horizon Optimization of Urban Traffic ControlBased on an Ant Algorithm [J]. *Control and Decision*, 2004, 19(9): 1057-1063.)

[6] 闻育. *复杂多阶段运动决策的蚁群优化方法及其在交通控制系统中的应用*[D]. 杭州: 浙江大学, 2004.
(Wen Y. *Ant Coloy Optimization for Complex Multi-stage Dynamic Decision Making and Its Applications in Traffic System Control* [D]. Hangzhou: Zhejiang University, 2004.)

[7] Snir M, Otto S, Huss-Lederman S. *MPI: The Complete Reference*[M]. London: MIT Press, 1998.

[8] 郁志辉. *高性能并行技术——MPI 并行程序设计*[M]. 北京: 清华大学出版社, 2001.
(Yu Z H. *High Perfomance Parallel Programming: MPI*[M]. Beijing: Tsinghua University Press, 2001.)

(下转第 1223 页)

而

$$\bar{F}_{11} = \mathrm{diag}(F_{11}, F_{13}, F_{31}, F_{33}),$$
$$\bar{F}_{12} = \mathrm{diag}(F_{12}, F_{32}),$$
$$\bar{F}_{21} = \mathrm{diag}(F_{21}, F_{23}),\ \bar{F}_{22} = F_{22},$$
$$\bar{B}_0 = \begin{bmatrix} I & 0 \\ 0 & B_0 \end{bmatrix},\ \bar{C}_0 = \begin{bmatrix} I & 0 \\ 0 & C_0 \end{bmatrix}.$$

由假设(5), 显然 $\bar{F}_{ij}^{\mathrm{T}}\bar{F}_{ij} \leq I, i, j = 1, 2$

另一方面, 系统(16) 与控制器(26) 构成的闭环系统为

$$\dot{\eta} = \bar{A}\eta + \begin{bmatrix} \\ B_\epsilon \\ \\ B_0 D_{21\epsilon} \end{bmatrix} \tilde{w},$$
$$\tilde{z} = [C_\epsilon \ D_{12\epsilon} C_0]\eta + D_\epsilon \tilde{w}.$$

注意到(22)～(25) 和 $B_{1\epsilon}, C_{1\epsilon}, D_{12\epsilon}, D_{21\epsilon}, D_\epsilon$ 的定义, 根据定理 1, 定理 2 即得证 □

由定理 2, 不确定系统的鲁棒正实控制问题可转化为确定系统的正实控制问题, 而后者可用 Riccati 方法或线性矩阵不等式方法求解[4]. 因此, 定理 2 给出了输出反馈鲁棒正实控制问题的一种处理方法

5　结　　语

对参数独立摄动不确定线性系统, 研究了鲁棒正实性分析和设计问题. 通过构造增广系统将不确定系统的鲁棒正实性分析和设计转化为确定系统的情形, 导出了系统鲁棒正实性条件, 给出了输出反馈鲁棒正实控制问题的处理方法. 由于参数独立摄动不确定性包括了通常的范数有界不确定性, 所得结果将现有鲁棒正实控制问题的相关结果推进了一步.

参考文献(References)

[1] Anderson B D O, Vangpanitlerd S. *Network Analysis and Synthesis: A Modern Systems Theory Approach* [M]. England Cliffs: Prentice-Hall, 1973.

[2] Vidyasagar M. *Nonlinear Systems Analysis* [M]. England Cliffs: Prentice-Hall, 1993.

[3] 王龙, 黄琳. 区间有理函数严格正实性的有限检验[J]. *科学通报*, 1991, 36(4): 262-264.
(Wang L, Huang L. Finite Strictly Positive Realness Test of Interval Rational Functions[J]. *Chinese Science Bulletin*, 1991, 36(4): 262-264.)

[4] Molander P, Willems J C. Synthesis of State Feedback Control Laws with a Specified Gain and Phase Margin [J]. *IEEE Trans on Automatic Control*, 1980, 25(7): 928-931.

[5] Sun W, Khargoneckar P, Shim D. Solution to the Positive Real Control Problem for Linear Time-invariant Systems[J]. *IEEE Trans Automatic Control*, 1994, 39(10): 2034-2046.

[6] 郭雷, 忻欣, 冯纯伯. 线性对象的正实控制问题[J]. *自动化学报*, 1997, 23(5): 577-582.
(Guo L, Xin X, Feng C B. The Positive Real Control Problem for Generalized Linear Plants [J]. *Acta Automatica Sinica*, 1997, 23(5): 577-582.)

[7] 邵汉永, 冯纯伯. 一类不确定多变量系统的鲁棒严格正实性分析及其输出反馈控制[J]. *东南大学学报*, 2003, 33(4): 492-494.
(Shao H Y, Feng C B. Robustly Strict Positive Real Analysis and Output Feedback Control for a Class of Uncertain MIMO Linear Systems [J]. *J of Southeast University*, 2003, 33(4): 492-494.)

[8] 邵汉永, 冯纯伯. 严格正实线性多变量系统的鲁棒性分析及其输出反馈控制 [J]. *控制与决策*, 2004, 19(3): 277-280.
(Shao H Y, Feng C B. Robustness Analysis and Feedback Control for Strictly Positive Real Linear MIMO Systems[J]. *Control and Decision*, 2004, 19(3): 277-280.)

[9] Xie L, Soh Y C. Positive Real Control Problem for Uncertain Linear Time-invariant Systems [J]. *Systems and Control Letters*, 1995, 24(4): 265-271.

[10] Mahmoud M S, Xie L. Positive Analysis and Synthesis for Uncertain Discrete-time Systems [J]. *IEEE Trans on Circuits System I*, 2000, 47(3): 403-406.

[11] Mahmoud M S, Soh Y C, Xie L, et al. Observed-based Positive Real Control of Uncertain Linear Systems[J]. *Automatica*, 1999, 35(4): 749-754.

(上接第 1218 页)

[9] Marcus R. A Parallel Implementation of Ant Colony Optimization [J]. *Parallel and Distributed Computing*, 2002, 62: 1421-1432.

[10] Stutzle T. Parallelization Strategies for Ant Colony Optimization [A]. *Proc of Parallel Problem Solving from Nature* 1998 [C]. Berlin: Springer-Verlgy, 1998: 722-741.