竞争决策算法及其在车辆路径问题中的应用①

宁爱兵, 马 良
(上海理工大学管理学院, 上海 200093)

摘要: 在分析自然界各种竞争机制和人类社会决策原理的基础上,利用竞争造就优化和决策左右结果的特性,提出了一种能广泛应用于组合优化难题的新型算法——竞争决策算法(CDA),并给出了 CDA 的通用模型. 车辆路径问题(VRP)是一个著名的 NP 难题,也是物流领域内一个重要的调度问题,利用 CDA 的通用模型设计了一个针对 VRP 的快速求解算法,并用该算法求解了 VRP 标准测试库中的实例,经过大量数据测试和验证,获得了令人满意的效果,其中部分问题的解优于目前公布的最好解.

关键词: 竞争决策算法; 竞争力函数; 决策函数; 车辆路径问题
中图分类号: O223 **文献标识码**: A **文章编号**: 1007 - 9807(2005)06 - 0010 - 09

0 引 言

计算机是人类解决复杂性问题的一个新的有力工具,人类对它的依赖性也越来越强. 但由于现实世界中很多问题都是高度非线性的复杂问题,人类对它们的研究还不够深入,因此仍然还有许多复杂的问题得不到很好的解决,如问题难度随着问题规模指数增长的 NP(non-deterministic polynomial time,非确定多项式时间算法)难题到目前还没有很好的解决方法[1,2]. 大自然是人类获得灵感的源泉,人们在自然界的启示下,将自然界所提供的机制和特性应用于实际问题求解已被证明是一种成功的方法,如计算机界模拟自然界的一些机制和特性提出了诸如遗传算法[1,3]、模拟退火算法[1,3]、免疫算法[4,5]、蚂蚁算法[6~12]、鱼群算法[13]、量子算法等,这些算法能对一些问题起到优化作用,但也存在局限性. 本文在分析大自然生物世界特别是人类的各种竞争机制和决策原理的基础上,利用竞争造就优化和决策左右结果的特性,提出了一种能广泛应用于组合优化问题的新型算法——竞争决策算法(competitive decision algorithm, CDA). 文中给出了该算法的基本概念及含义、通用模型和算法的特点,并利用该通用模型设计了一个针对车辆路径问题的快速求解算法,经过大量数据测试和验证,并与这些问题的已有结果进行比较,获得了令人满意的效果,其中部分问题的解优于目前公布的最好解.

1 竞争决策算法原理

1.1 竞争造就优化、决策左右结果

达尔文的"适者生存"说明自然界中的生物世界通过竞争能够实现优胜劣汰的效果,竞争能够保留具有优势的合理因素而淘汰那些具有劣势的不合理因素,这说明竞争能够起到优化的作用;决策是人们在政治、经济、技术和日常生活中普遍存在的一种选择方案的行为,在竞争过程中,竞争各方要根据各种条件采用合理的对策,竞争各方的决策会影响到最终的竞争结果,我国古代著名的"田忌赛马"就是一个很好的例子,这说明竞争各方的决策在很大程度能决定竞争的结果. 这种竞争造就优化和决策左右结果的特性在现实世界中

① 收稿日期: 2004 - 11 - 25; 修订日期: 2005 - 09 - 28.
基金项目: 国家自然科学基金资助项目(70471065).
作者简介: 宁爱兵(1972—), 男, 湖南邵东人, 博士生.

随处可见,如在合理竞争和决策规则下的市场经济竞争能够合理地分配人力和物质资源,使各种资源能够得到优化配置而提高经济效益;合理决策使一批公司得到壮大和发展,不合理决策使一批公司削弱,甚至消亡. 合理竞争和决策还能使缺乏控制中心的系统高效有序的运行,如纽约、上海和东京这样的大城市在合理商品竞争和决策规则下既没有一个统一控制中心来安排购买和配送,也没有保持大量的储备来发生缓冲作用以便对付市场波动,就能够不间断的保障食品、医疗和数百万居民必需品的供给[14~16].

1.2　竞争决策算法的实现

自然界中的竞争和决策都是在一定竞争规则下,在竞争者的实力、竞争者和环境间的关系、多个竞争者实力的差距和初始竞争格局等多种因素的作用下,经过多次竞争和决策后,使不同的竞争者分别占有一定的资源而达到一种新的竞争格局,只要新的竞争格局优于初始竞争格局,就会达到优化的目的. CDA 就是利用该原理来达到优化的目的,它通过构造一个或多个竞争者参与到对一个或多个资源的竞争的过程中,通过优胜劣汰的决策原则使一部分竞争者获得资源而增加实力,使另一部分竞争者失去资源而减弱实力甚至死亡. 对 CDA 的结果有较大影响的因素有以下几种:

(1) 竞争规则(Competition rules)

竞争规则是竞争决策算法中竞争者必须要满足的条件,它包括以下 3 种条件:

初始条件(又称前置断言 precondition):在开始进行竞争时必须满足的条件.

竞争条件(又称竞争不变式 competition_Invariants):在竞争过程中始终必须满足的条件.

终止条件(又称后置断言 postcondition):在竞争过程中若满足该条件就可以终止当前的一次竞争和决策,此时竞争者各方达到一个竞争平衡状态,即此时每个竞争者都不能凭自己的实力从竞争对手手中抢占资源.

(2) 竞争力函数(competitive force function)

竞争力函数代表竞争者对某种资源具有的竞争力,一般来说,竞争力函数大的竞争者占有该资源的可能性要大.

(3) 决策函数(decision function)

决策函数起到裁判的作用,它根据每个竞争者竞争力函数值的大小来决定把资源分配给哪一个竞争者,如可将决策函数定义成把某资源分配给对该资源具有最大竞争力的竞争者.

(4) 初始格局(initial layout)

初始格局是指在某一轮竞争开始前各个竞争者对资源的占有情况和竞争者的实力情况.

这些因素中的某一个单独发生变化或者组合发生变化都会对竞争和决策的结果产生较大的影响,对同一个问题来说竞争规则一般只有一个,为了取得较好的效果,可能对同一个问题采用多个竞争力函数、多个决策函数和多个初始格局的组合来实现 m(m = 竞争力函数个数 ×决策函数个数 ×初始格局个数)轮竞争,最后取竞争和决策效果最好的值作为优化值.

1.3　竞争决策算法通用流程

步骤 1　初始化

if　满足初始条件　then

{最大的竞争步数 = 某个与问题规模成正比的常数

p_count = 竞争力函数个数; d_count = 决策函数个数; la_count = 初始格局个数}

else {退出程序}

步骤 2　竞争与决策

for p = 1 to p_count　// 竞争力函数个数循环

for d = 1 to d_count　// 决策函数个数循环

for la = 1 to la_count　// 初始格局个数循环

{进行一些初始化工作,为竞争做准备

根据 la 设置初始格局

竞争步数 = 0

根据当前 p 计算在初始格局下每个竞争者对每个资源具有的竞争力

repeat // repeat 与 until 之间的操作循环

根据当前 d 决定某个竞争者占有某个资源而产生新的格局

根据当前 p 重新计算在新格局下每个竞争者对每个资源具有的竞争力

(注意:只有当某个竞争者对某个资源的竞争力发生变化时才重新计算)

竞争步数 = 竞争步数 + 1

until (当前格局满足后置断言) or (竞争步数 > 最大的竞争步数)

if 当前格局满足终止条件

把本轮竞争和以前的竞争结果进行比较并保存好的结果

}

步骤 3 输出竞争得到的最好的一个或几个结果.

该算法共竞争 m ($m = p\_count \times d\_count \times la\_coun$) 轮,每一轮竞争得到一个满足终止条件的竞争格局,竞争步数记数是为了在得不到满足终止条件的情况下终止本轮竞争并放弃本轮竞争结果,而最后的结果为所有结果中最好的值,该算法在最好情况下的时间复杂度为 O($p_count \times d_count \times la_count \times$满足终止条件时的资源个数),在最坏情况下为 O($p_count \times d_count \times la_count \times$最大的竞争步数).

1.4 竞争决策算法的应用

得出 CDA 通用流程后,对于不同的问题,只要设置好竞争规则、竞争力函数、决策函数和初始格局后就能得到一个求解该问题的 CDA,并且可以通过修改、新增竞争规则、竞争力函数、决策函数、初始格局来优化求解的问题;同时,如果我们把竞争规则、竞争力函数、决策函数和初始格局存放到数据库中供用户随时增添修改,就能通过 CDA 通用流程来实现该算法的自动化和半自动化,用户只需要增添修改竞争规则、竞争力函数、决策函数和初始格局就能得到一个新的算法和新的解. CDA 本身是确定性的,但当竞争力函数、决策函数、初始格局等细节部分含有随机因子并且随机因子对结果产生了影响,CDA 也会因此变为随机算法.

CDA 能广泛应用于组合优化问题,我们已经利用 CDA 通用流程实现了车辆路径问题 (vehicle routing problem, VRP)、旅行商问题 (traveling salesman problem, TSP) 和度约束最小生成树 (degree-constrained minimum spanning tree, DCMST) 等 NP 难题的算法,这里只介绍 VRP 的 CDA,其它问题的算法将陆续成文.

2 车辆路径问题(VRP) 的竞争决策算法

2.1 问题描述

VRP 是物流管理中的一个重要 NP 难题,一般意义下的 VRP 可描述如下:在约束条件下,设计从一个或多个初始点出发,到多个不同位置的城市或客户点的最优送货巡回路径,即设计一个总耗费最小的路线集,使其满足:

(1) 每个城市或客户只被一辆车访问一次.

(2) 所有车辆从起点出发再回到起点.

(3) 某些约束被满足.

最通常的约束包括容量限制 (capacitated VRP, CVRP)、时间窗限制 (VRP with time windows, VRPTW) 等. CVRP 只有一个固定的出发点并且该结点的编号为 1,它要受到容量的限制,并且规定每辆车的容量相等,即任何一辆车在行驶路径上所提供的货物总量不能超出车辆的装载能力[17~23]. 本文的 VRP 算法是针对 CVRP 来设计的,通过修改竞争规则、竞争力函数、决策函数和初始格局中的一项或几项就能得到求解其它 VRP 问题的算法.

2.2 基本思想

CVRP 竞争决策算法的基本思想如下:

(1) 把 CVRP 问题分为 2 步,第 1 步确定所需车辆数及每辆车需要访问的结点,第 2 步确定每辆车所访问结点的 TSP 回路.

(2) 在第 1 步中把每辆车当作一个竞争者,每个结点当作被竞争的资源,若车没有占有结点或只占有结点 1,则认为该车被淘汰. 由于结点 1 是固定的出发点,因此它属于每一个竞争者且不被当作资源来竞争.

(3) 在第 2 步中,由于每辆车要访问的结点已确定,因此实际上是一个 TSP 问题,解决该问题的方法很多,本文采用分枝定界法 (Branch and Bound)[24] 和蚂蚁算法[6~10] 来求解.

2.3 基本符号及含义

为了方便描述,采用如下符号表示:

n ——CVRP 问题结点总个数.

$W[i,j]$ ——结点 i 到结点 j 之间的距离, $W[i,i] = \infty$.

$Capacity$ ——每辆车的最大装载容量.

$demand[j]$ ——结点 j 所需商品的数量.

$cur\_capacity[j]$ ——车辆在装载某些访问的结点的货物后剩下的装载容量.

$owner[j]$ ——访问结点 j 的车辆编号.

$all\_power[i,j]$ ——车辆 i 对结点 j 所具有的竞争力.

$max\_power[j,1]$ ——所有车辆中对结点 j 具有的最大竞争力.

$max\_power\_id[j,1]$: 对结点 j 具有最大竞争力的车辆号.

max_power[j,2] ——所有车辆中对结点 *j* 具有的次大竞争力.

max_power_id[j,2] ——对结点 *j* 具有次大竞争力的车辆号.

min_len[i,j,1] ——车辆 *i* 已访问的所有结点到结点 *j* 的最小距离.

min_len[i,j,2] ——车辆 *i* 已访问的所有结点到结点 *j* 的次小距离.

min_len[i,j,3] ——车辆 *i* 已访问的所有结点到结点 *j* 的第 3 小距离.

min_len[i,j,4] ——车辆 *i* 已访问的所有结点到结点 *j* 的第 4 小距离.

2.4　竞争规则

采用以下竞争规则:

(1) 初始条件:每个结点所需商品数量不能超过车辆的最大装载容量.

(2) 竞争条件:在竞争过程中每辆车经过结点所需商品数量之和不会超过车辆总容量之和.

(3) 终止条件:在后面的算法流程中介绍.

2.5　竞争力函数和决策函数

采用以下 6 个竞争力函数(目的是为了使一辆车尽量访问地理位置靠近的点):

(1) *all_power[i,j]* = 1/ *min_len[i,j,1]* + 1/ *min_len[i,j,2]* (考虑了最短的 2 条边)

(2) *all_power[i,j]* = 1/ *min_len[i,j,1]* (考虑了最短的 1 条边)

(3) *all_power[i,j]* = 1/ *min_len_1[i,j,1]* + 1/ *min_len[i,j,2]* + 1/ *min_len[i,j,3]* (考虑了最短的 3 条边)

(4) *all_power[i,j]* = 1/(*min_len[i,j,1]* + *min_len[i,j,2]*) (考虑了最短的 2 条边)

(5) *all_power[i,j]* = 1/(*min_len[i,j,1]*/2 + *min_len[i,j,2]*) (考虑了最短的 2 条边)

(6) *all_power[i,j]* = 1/ *min_len[i,j,1]* + 1/ *min_len[i,j,2]* + 0.75/ *min_len[i,j,3]* + 0.5/ *min_len[i,j,4]* (考虑了最短的 4 条边并设置了相应的权值)

(注:以上所有竞争力函数中,若 *cur_capacity[i]* < *demand[j]* 且 *owner[j]* < > *i* 则置 *all_power[i,j]* = - ∞,其目的是保证车辆经过结点所需商品数量之和不会超过车辆总容量.)

采用的决策函数(有贪心的和不贪心的决策函数)在算法流程中介绍.

2.6　初始格局

由于开始时,每辆车都占有固定出发点 1,此时每辆车(共有 *n* - 1 辆车)对每个结点的竞争力都相同,并且每辆车的容量相等,因此第一次把结点 *i* 分配给任意一辆车都没有区别,对结果有影响的是 *i* 的值,因此初始格局共有 *n* - 1 个.

3　CVRP 的竞争决策算法流程

步骤 1　初始化

if　满足初始条件 then

{最大的竞争步数 = 20 * n

p_count = 6; *d_count* = 5; *la_count* = *n* - 1 //此三项分别为竞争力函数、决策函数和初始格局的个数}

else {否则退出程序}

步骤 2　竞争与决策

for *p* = 1 to *p_count* // 竞争力函数个数循环

for *d* = 1 to *d_count* // 决策函数个数循环

for *la* = 1 to *la_count* // 初始格局个数循环

{for *i* = 2 to *n* // 结点 1 为固定结点,故共有 *n* - 1 个结点要被 *n* - 1 辆车占有

{ *cur_capacity[i]* = *capacity* // 每一辆车的剩余转载容量为 *capacity*

owner[i] = - 5 // 结点 *i* 没有被车占有,即结点 *i* 还没有被某一辆车竞争到

}

owner[la + 1] = 2 // 结点(*la* + 1) 被车辆 2 占领形成初始格局

cur_capacity[2] = *cur_capacity*[2] - *demand*[*la* + 1] // 调整车辆 2 的剩余转载容量

计算 *min_len[i,j,1]*, *min_len[i,j,2]*, *min_len[i,j,3]*, *min_len[i,j,4]*

根据第 p 个竞争力函数计算 *all_power[i,j]*

根据 *all_power[i,j]* 计算 *max_power[j,k]* 和 *max_power_id[j,k]* (这里的 2 ≤*i* ≤*n*,2 ≤*j* ≤*n*,1 ≤*k* ≤2)

竞争步数 = 0

repeat // repeat 与 until 之间的操作循环

cur_max_supp = - ∞;

change_id = 0; // *change_id* 保存本次竞争决策要改变其访问车辆的结点

select case *p* // 根据 p 来选择决策函数,以下为 5 个决策函数

case 1: // 决策函数 1

For *j* = 2 To *n* do // 第 j 点,

{if ((*max_power[j,1]* - *max_power[j,2]*) > *cur_max_supp*)

if (*owner[j]* < > *max_power_id[j,1]*)

and (j < > change_id)

{ cur_max_supp = max_support[j, 1] - max_support[j, 2]

change_id = j}}

(决策函数 1 的含义是选中(max_power[j, 1] - max_power[j,2])最大的结点 change_id(当两个点具有相同的值时,选编号小的),并使车辆 max_power_id[change_id, 1]访问结点 change_id,该函数不贪心)

case 2 //决策函数 2

For j = 1 To n // 第 j 点,

{If (max_power[j, 1] > cur_max_supp)

If (owner[j] < > max_power_id[j, 1]) and (j < > change_id)

{ cur_max_supp = max_power[j, 1]

change_id = j}}

(决策函数 2 的含义是选中 max_power[j,1]最大的结点 change_id(当两个点具有相同的值时,选编号小的),并使车辆 max_power_id[change_id, 1]访问结点 change_id,该函数贪心)

case 3 //决策函数 3

把 case 1 中的第一个" > "改为" > = ",其它与 case 1 完全相同

(决策函数 3 的含义与 1 的不同之处在于当两个点具有相同的值时,选编号大的.)

case 4 决策函数 4

把 case 2 中的第一个" > "改为" > = ",其它与 case 2 完全相同

(决策函数 4 的含义与 2 的不同之处在于当两个点具有相同的值时,选编号大的.)

case 5 决策函数 5

temp_d = 0 到 1 之间的一个随机数

if temp_d < 0.5 {与 case 3 的完全相同}

else { 与 case 1 的完全相同}

end select

竞争步数 = 竞争步数 + 1

old_owner = - 5

if change_id > 1 // 找到符合要求的结点 change_id

{ old_owner = owner[change_id] // old_owner 为原来访问结点 change_id 的车辆编号

cur_capacity[old_owner] = cur_capacity[old_owner] + demand[change_id] }

i = max_power_id[change_id, 1] // 原来被车辆 owner[change_id]访问的结点现在改为由车辆 max_power_id[change_id, 1] 访问

owner[change_id] = i

cur_capacity[i] = cur_capacity[i] - demand[change_id]// 调整车辆 i 的剩余转载容量

重新计算第 i 辆车对所有结点的 min_len[i, j, k],其中 2 ≤ j ≤ n, 1 ≤ k ≤4

根据第 p 个竞争力函数计算第 i 辆车对所有结点的竞争力 all_power[i, j]

if old_owner > 0 then // 结点 change_id 原来被车辆 old_owner 访问

{重新计算第 old_owner 辆车对所有结点的 min_len[old_owner, j, k]

根据第 p 个竞争力函数' 计算第 old_owner 辆车对所有结点的竞争力 all_power[old_owne, j]}

(注意:上面 2 ≤ j ≤ n, 1 ≤ k ≤4,不需要更新其它车辆对结点的竞争力,因为它们没有发生变化)

根据新的 all_power[i, j]计算 max_power[j, k]和 max_power_id[j, k] (这里的 2 ≤ i ≤ n,2 ≤ j ≤ n,1 ≤ k ≤2)

until (change_id < 1 or 竞争步数 > 最大的竞争步数)

(change_id < 1 意味着按照决策函数,找不到一辆车能够把一个结点从另一辆车手中抢占过来,这说明此时的竞争达到一个平衡状态.)

if 除结点 1 外每个结点都被且只被一辆车访问一次

for i = 2 to n // 处理每辆车

if 车辆 i 访问的结点数 > = 2 // 淘汰访问结点个数小于 2 的车辆

if 车辆 i 访问的结点数 < = 25

{调用分枝定界法[24]求车辆所访问结点的 TSP 回路并累加其回路长度}

else

{调用蚂蚁算法[6~12]求车辆 i 所访问结点的 TSP 回路并累加其回路长度}

把本轮竞争决策结果和以前的竞争结果进行比较并保存好的结果(累加的回路长度为 VRP 长度)

}

步骤 3 输出竞争得到的最好的一个或几个结果.

不难估算出,该算法在最好情况下的时间复杂度为 O(6 ×5 ×(n - 1) ×(n - 1) ×2 ×(n - 1)),在最坏情况下的时间复杂度为 O(6 ×5 ×(n - 1) ×最大的竞争步数 ×2 ×(n - 1)),其中的 2 ×(n - 1)表示当一辆车所访问的结点发生变化时,要重新计算 2 辆车对其它(n - 1)个结点的竞争力.

4　实例计算

为检验算法效果,用该算法求解了“http://neo.lcc.uma.es/radi-aeb/WebVRP/index.html?/algorithms/ClustRout.html/”中公布的 CVRP 问题库中的部分实例,表 1 到表 6 给出了求解的结果并计算了有关参数(在计算边长时四舍五入取整),其中 n 为结点总个数, *Capacity* 为每辆车的最大装载容量,$L1$ 为本算法求得的解,$L2$ 为问题库中公布的最好解的长度,在当前最好解的基础上改进的百分比 $= (L2 - L1)/L2$, $b1 = (L1 - L2)/L2$,这里把本算法求解结果小于公布的最好解的一部分解给出了详细情况(见表 1 到表 5),其中 V 为车辆编号,V_L、V_C 和 V_R 分别为对应车辆的回路长度,运输量和回路路径,其他问题只给出结果和参数 $b1$(见表 6).

问题 1　(库文件名:A-n60-k9,其中 $n = 60$, *Capacity* = 100, $L2 = 1\ 408$) 本算法求解的结果为:

所需车辆数 = 9, 车辆总行程 = 1 394(在当前最好解的基础上改进的百分比 = 1 %).

表 1　问题 1 解的详细情况

Table 1 The particular result of problem 1

V	V_L	V_C	V_R
1	90	87	1,19,20,60,39,53,34,42,1
2	198	87	1,43,49,37,2,1
3	162	92	1,51,40,27,18,38,58,28,1
4	123	97	1,15,48,24,59,25,3,35,1
5	230	93	1,44,57,13,52,10,33,1
6	129	100	1,36,56,16,9,14,30,8,1
7	182	94	1,23,11,55,6,46,1
8	177	95	1,7,29,45,50,31,54,32,1
9	103	84	1,17,21,4,12,5,22,41,47,26,1

问题 2　(库文件名:B-n66-k9,其中 $n = 66$, *Capacity* = 100, $L2 = 1\ 374$) 本算法求解的结果为

所需车辆数 = 9, 车辆总行程 = 1 349(在当前最好解的基础上改进的百分比 = 1.8 %).

表 2　问题 2 解的详细情况

Table 2 The particular result of problem 2

V	V_L	V_C	V_R
1	168	98	1,51,29,7,3,45,8,1
2	204	90	1,10,37,25,13,50,1
3	104	100	1,56,12,58,39,24,19,1
4	115	100	1,46,32,14,57,23,44,35,52,1
5	46	96	1,2,18,21,54,55,53,42,1
6	147	97	1,31,62,38,26,49,63,17,48,1
7	251	94	1,33,59,41,5,16,61,22,4,30,40,15,1
8	169	98	1,20,36,60,65,64,47,27,11,1
9	145	88	1,34,9,66,6,28,43,1

问题 3　(库文件名:P-n20-k2,其中 $n = 20$, *Capacity* = 160, $L2 = 220$) 本算法求解的结果为

所需车辆数 = 2, 车辆总行程 = 217(在当前最好解的基础上改进的百分比 = 1.4 %).

表 3　问题 3 解的详细情况

Table 3 The particular result of problem 3

V	V_L	V_C	V_R
1	89	155	1,7,3,8,10,17,15,6,20,1
2	128	155	1,2,11,14,9,18,19,4,13,16,12,5,1

问题 4　(库文件名:P-n22-k8,其中 $n = 22$, *Capacity* = 3 000, $L2 = 603$) 本算法求解的结果为

所需车辆数 = 9, 车辆总行程 = 600(在当前最好解的基础上改进的百分比 = 0.5 %).

表 4　问题 4 解的详细情况

Table 4 The particular result of problem 4

V	V_L	V_C	V_R
1	22	2 400	1,15,17,1
2	43	2 200	1,13,16,1
3	56	2 500	1,14,12,1
4	89	2 900	1,11,9,4,5,1
5	62	2 500	1,20,1
6	76	2 600	1,10,6,1
7	70	1 700	1,18,22,1
8	109	3 000	1,7,2,3,8,1
9	73	2 700	1,19,21,1

问题 5　(库文件名:B-n51-k7,其中 $n = 51$, *Capacity* = 100, $L2 = 1\ 032$) 本算法求解的结果为

所需车辆数 = 8, 车辆总行程 = 1 019(在当前最好解的基础上改进的百分比 = 1.3 %).

表 5 问题 5 解的详细情况

Table 5 The particular result of problem 5

V	V_L	V_C	V_R
1	94	42	1 ,19 ,1
2	114	88	1 ,7 ,8 ,15 ,29 ,27 ,44 ,1
3	152	100	1 ,9 ,13 ,38 ,33 ,32 ,28 ,42 ,1
4	113	99	1 ,40 ,30 ,14 ,43 ,5 ,12 ,1
5	148	72	1 ,36 ,4 ,20 ,34 ,49 ,1
6	141	92	1 ,24 ,46 ,48 ,31 ,25 ,23 ,47 ,26 ,1
7	135	100	1 ,22 ,45 ,21 ,2 ,10 ,50 ,16 ,51 ,1
8	122	91	1 ,6 ,37 ,41 ,17 ,3 ,39 ,35 ,11 ,18 ,1

其它问题的求解结果只给出结果和参数 $b1$ (见表 6) :

表 6 计算结果

Table 6 The computational result

编号	问题名称	$L1$	$L2$	$b1$
1	A-n32-k5	806	784	0.028
2	A-n33-k5	682	661	0.032
3	A-n33-k6	743	742	0.001
4	A-n34-k5	797	778	0.024
5	A-n36-k5	808	799	0.011
6	A-n37-k5	731	669	0.093
7	A-n37-k6	966	949	0.018
8	A-n38-k5	750	730	0.027
9	A-n39-k5	858	822	0.044
10	A-n39-k6	855	831	0.029
11	A-n44-k7	962	937	0.027
12	A-n45-k6	968	944	0.025
13	A-n45-k7	1 208	1 146	0.054
14	A-n46-k7	970	914	0.061
15	A-n48-k7	1 117	1 073	0.041
16	A-n53-k7	1 057	1 010	0.047
17	A-n54-k7	1 222	1 167	0.047
18	A-n55-k9	1 088	1 073	0.014
19	A-n61-k9	1 098	1 034	0.062
20	A-n62-k8	1 352	1 290	0.048
21	A-n63-k10	1 379	1 315	0.049
22	A-n63-k9	1 698	1 634	0.039

续表 6

编号	问题名称	$L1$	$L2$	$b1$
23	A-n64-k9	1 473	1 402	0.051
24	A-n65-k9	1 185	1 174	0.009
25	A-n69-k9	1 220	1 168	0.045
26	A-n80-k10	1 838	1 764	0.042
27	B-n31-k5	677	672	0.007
28	B-n34-k5	788	788	0.000
29	B-n35-k5	962	955	0.007
30	B-n38-k6	819	805	0.017
31	B-n39-k5	566	549	0.031
32	B-n41-k6	836	829	0.008
33	B-n43-k6	748	742	0.008
34	B-n44-k7	940	909	0.034
35	B-n45-k5	763	751	0.016
36	B-n45-k6	735	678	0.084
37	B-n50-k7	759	741	0.024
38	B-n50-k8	1 346	1 313	0.025
39	B-n52-k7	758	747	0.015
40	B-n56-k7	730	707	0.033
41	B-n57-k7	1 155	1 153	0.002
42	B-n57-k9	1 636	1 598	0.024
43	B-n63-k10	1 565	1 537	0.018
44	B-n64-k9	906	861	0.052
45	B-n67-k10	1 089	1 033	0.054
46	B-n68-k9	1 311	1 304	0.005
47	B-n78-k10	1 276	1 266	0.008
48	E-n101-k14	1 157	1 077	0.074
49	E-n101-k8	889	817	0.088
50	E-n22-k4	376	375	0.003
51	E-n23-k3	569	569	0.000

5 结束语

利用 CDA 通用流程,只要分别设计各问题的竞争规则、竞争力函数、决策函数和初始格局就能求解一系列组合优化难题,其通用性和实用性都比较强,适应范围也较广,已经实现了 VRP、TSP 和 DCMST 等 NP 难题的 CDA 算法,其效果都比较好,本文给出的 VRP 竞争决策算法就是一个很好的例子. 随着研究的进一步深入,相信 CDA 的应用前景和应用领域将非常广阔.

参 考 文 献:

[1]王正志, 薄　涛. 进化计算[M]. 长沙: 国防科技大学出版社, 2000. 1—40.
Wang Zhengzhi, Bo Tao. Evolutionary Computation[M]. Changsha: Press of National University of Defense Technology, 2000. 1—40. (in Chinese)

[2]潘正君, 康立山, 陈毓屏. 演化计算[M]. 北京: 清华大学出版社, 1998. 1—8.
Pan Zhengjun, Kang Lishan, Chen Yuping. Evolutionary Computation[M]. Beijing: Tsinghua University Press, 1998. 1—8. (in Chinese)

[3]Holland J H. Genetic algorithms and classifier systems: Foundations and Their Application[C]. Proc of the Second Int Conf on Genetic Algorithms, 1987. 82—89.

[4]王　磊, 潘　进, 焦李成. 免疫算法[J]. 电子学报, 2000, 28 (7): 74—78.
Wang Lei, Pan Jin, Jiao Licheng. The immune algorithm[J]. Acta Electronica Sinica, 2000, 28 (7). 74—78. (in Chinese)

[5]Chun Jang-Sung, Jang Hyun-Kyo, Hahn Song-Yop. A study on comparsion of optimization performances between immune algorithm and other heuristic algorithms[J]. IEEE Trans On Magnetics, 1998. 34 (5): 2972—2975.

[6]Colorni A, Dorigo M, Maniezzo V. An investigation of some properties of an ant algorithm[A]. Proc of the Parallel Problem Solving from Nature Conf (PPSN '92)[C]. Brussels, Belgium: Elsevier Publishing, 1992. 509—520.

[7]Dorigo M, Maniezzo V, Colorni A. Ant system: Optimization by a colony of cooperation agents[J]. IEEE Transactions on Systems, Man, and Cybernetics, 1996, 26 (1): 29—41.

[8]Colorni A, Dorigo M, Maniezzo V, *et al*. Ant system for job-shop scheduling[J]. Journal of Operations Research, 1994, 34 (1): 39—53.

[9]马　良, 项培军. 蚂蚁算法在组合优化中的应用[J]. 管理科学学报, 2001, 4 (2): 32—37.
Ma Liang, Xiang Peijun. Applications of ant algorithm to combinatorial optimization[J]. Journal of Management Sciences in China, 2001, 4 (2): 32—37. (in Chinese)

[10]马　良. 全局优化的一种新方法[J]. 系统工程与电子技术, 2000, 22 (9): 61—62.
Ma Liang. A new Method for Global Optimization[J]. Systems Engineering and Electronics, 2000, 22 (9): 61—62. (in Chinese)

[11]Ma Liang, Wang Longde. Artificial Ant Algorithm for Constrained Optimization[J]. Journal of Systems Science and Systems Engineering, 2001, 10 (1): 57—61.

[12]Ma Liang, Yao Jian. A new algorithm for integer programming problem[A]. Proc of 2001 Int. Conf. on Management Science & Engineering[C], Harbin Institute of Technology Press, 2001. 534—537.

[13]李晓磊, 邵之江, 钱积新. 一种基于动物自治体的寻优模式: 鱼群算法[J]. 系统工程理论与实践, 2002, 22 (11): 32—38.
Li Xiaolei, Shao Zhijiang, Qian Jixin. An optimizing method based on autonomous animats: Fish-swarm algorithm[J]. Systems Engineering-Theory Methodology Applications, 2002, 22 (11): 32—38. (in Chinese)

[14]约翰 H. 隐秩序 ——适应性造就复杂性[M]. 上海: 上海科技教育出版社, 2000. 1—3
John H. Hidden Order: How Adaptation Builds Complexity[M]. Reading: Addison-Wesley, 1996. 1—3.

[15]约翰 B, 戴维 F D. 混沌七鉴 ——来自易学的永恒智慧[M]. 上海: 上海科技教育出版社, 2001. 1—29.
John B, David F D. Seven Life Lessons of Chaos: Spiritual Wisdom from the Science of Change[M]. New York: Harpercollins, 1999. 1—29.

[16]钱颂迪. 运筹学[M]. 北京: 清华大学出版社, 1990. 388—440.
Qian Songdi. Operational Research[M]. Beijing: Tsinghua University Press, 1990. 388—440. (in Chinese)

[17]Laporte G. The vehicle routing problem: An overview of exact and approximation algorithms[J]. European Journal of Operational Research, 1992, 5 (9): 345—358.

[18]Gendreau M, Hertz A, Laporte G A. Tabu search heuristic for the vehicle routing problem[J]. Management Science, 1994, 40 (10): 1276—1290.

[19]汪寿阳, 赵秋红, 夏国平. 集成物流管理系统中定位 ——运输路线安排问题的研究[J]. 管理科学学报, 2000, 3(2):

69 —75.

Wang Shouyang , Zhao Qiuhong , Xia Guoping. Research on combined location-routing problems in integrated logistics systems[J]. Journal of Management Sciences in China , 2000 , 3(2) : 69 —75. (in Chinese)

[20] Gillett B E , Miller L R. A heuristic algorithm for the vehicle dispatch problem[J]. Operations Research , 1974 , 22 : 340—349.

[21] 谢秉磊 , 郭耀煌 , 郭 强. 动态车辆路径问题 : 现状与展望[J]. 系统工程理论方法应用 , 2002 , 11 (2) : 116 —120

Xie Binglei , Guo Yaohuang , Guo Qiang. Dynamic vehicle problems : Status and prospect[J]. Systems Engineering-Theory Methodology Applications , 2002 , 11(2) : 116 —120. (in Chinese)

[22] Rego C. A Sub path ejection method for the vehicle routing problem[J]. Management Science , 1998 , 44(10) : 1447 —1459.

[23] Savelsbergh M. Local search in routing problems with time windows[J]. Operations Research , 1985 , 33(4) : 285 —305.

[24] Syslo M M , Deo N , Kowalik J S. Discrete Optimization Algorithms[M]. Englewood Cliffs : Prentice-Hall , 1983. 346 —361.

Competitive decision algorithm and its application to vehicle routing problem

NING Ai-bing , MA Liang

College of Management , University of Shanghai for Science and Technology , Shanghai 200093 , China

Abstract : Through analysing the mechanism of natural competitions and the principle of decision , and based on the characteristics that competition builds optimisation and the result of competition hinges on decision , this paper proposes a new algorithm —Competitive decision algorithm , to solve combinatorial optimization problems. A general model for competitive decision algorithm is provided. And according to this model , a competitive decision algorithm for solving vehicle routing problem , which is a well-known NP-hard problem and is very important in practical transport logistics , is developed. By using the new algorithm , we solved some CVRP instances and compared these solutions with the best-known solutions. Computational results on benchmark problems show that this approach is promising in good performances and some solutions of the CVRP instances are better than the best solutions published.

Key words : competitive decision algorithm ; competitive force function ; decision function ; vehicle routing problem