一种新的进化粒子群算法及其在TSP中的应用

刘松兵,李智勇,王 永,孙星明
LIU Song-bing,LI Zhi-yong,WANG Yong,SUN Xing-ming

湖南大学 计算机与通信学院,长沙 410082
Department of Computer,College of Computer & Communication,Hunan University,Changsha 410082,China
E-mail:lzy_2000@sina.com

LIU Song-bing,LI Zhi-yong,WANG Yong,et al.Novel evolutionary Particle Swarm Optimization for Traveling Salesman Problem.Computer Engineering and Applications,2008,44(28):62-64.

Abstract: Inspired from the co-evolutionary,this paper proposes a new hybrid particle swarm evolutionary algorithm for solving the Traveling Salesman Problem (TSP),which is one of the most known NP hard problem.The algorithm adopts an effective code schema and defines a new addition operation of the particle's position in order to exchange information among the particles.A mutation operator is designed to keep the population's diversity.The experiments show that this algorithm has better convergence effectiveness.

Key words: particle swarm algorithm;evolutionary computation;Traveling Salesman Problem(TSP)

摘 要:基于协同进化的思想,针对离散组合优化的NP难问题,提出一种新的混合粒子群进化算法。该算法采用了有效的编码方式;定义了两个粒子间的位置加法操作以实现个体之间的信息交换;引入变异算子保持种群多样性。该算法应用于TSP优化计算,能用较小的计算代价得到比传统方法更满意的解,实验结果表明该算法是有效的。

关键词:粒子群算法;进化计算;旅行商问题
DOI:10.3778/j.issn.1002-8331.2008.28.022 **文章编号:**1002-8331(2008)28-0062-03 **文献标识码:**A **中图分类号:**TP391

1 引言

粒子群算法(Particle Swarm Optimization,PSO)是一种新兴的进化计算技术,最先由Kennedy和Eberhart提出[1],它的思想来源于鱼群和鸟群等社会群体生物的觅食现象。在PSO中,每一个粒子的位置代表问题的一个潜在解,在迭代过程中,每一个粒子跟随代表最优解的个体在解空间中进行搜索。粒子群算法简单容易实现,而且具有比普通遗传算法(Genetic Algorithm,GA)更高的效率,特别是在连续函数的优化方面,PSO算法表现出非常强的适应性,被广泛研究和采用。目前为止,PSO算法在离散参数优化方面的研究还不是很多,Kennedy和Eberhart提出的PSO算法的离散二进制版把粒子的位置表示为离散型的0和1,但速度还是连续型的量[2];在组合优化方面,Ching-JongLiao等用离散粒子群算法(Discrete Particle Swarm Optimization,DPSO)解决flow-shop问题[3],WANG Kangping等用PSO算法来求解旅行商问题(Traveling Salesman Problem,TSP)[4],等等。然而,已有的离散粒子群算法中都保留了粒子的速度这一概念,由于离散空间的特殊性,使得问题的求解变得相当复杂,而且算法的收敛效率不能令人满意。

Kennedy在文献[5]中已经说明速度方程不是粒子必须的,Jun Sun等受这一思想的影响,将取消了速度方程的量子粒子群优化算法应用在连续函数的优化方面取得了成功[6-7]。基于上述思想,本文提出了一种新的进化粒子群算法(Novel Evolutionary Particle Swarm Optimization,N-EPSO),结合一种适合TSP问题与本文算法的特殊编码方案,在此基础上新定义了两个粒子间的位置"加法"运算,替换原来的速度方程,以实现粒子间的信息交换,然后增加一个变异算子,用来保持种群的多样性,防止早熟收敛。为了分析算法的收敛效率,分别采用本文算法与文献[4]中的算法进行TSP优化实验,实验结果表明算法是有效的,其收敛效率高于参考算法。

2 粒子群优化算法

在PSO算法中,粒子群在一个 n 维的空间中搜索,其中每个粒子所处的位置都表示问题的一个潜在解。粒子通过不断调整自己的位置 X 来搜索新解。每个粒子都能记住自己搜索到的最好解,记作 P_{id},以及整个粒子群经历过的最好位置,即目前搜索到的最优解,记作 P_{gd}。每个粒子都有一个速度,记作 V,定义为:

基金项目:国家自然科学基金重点项目(the Key Project of National Natural Science Foundation of China under Grant No.60736016)。
作者简介:刘松兵(1984-),男,硕士研究生,主要研究领域为自然计算、人工智能;李智勇(1971-),男,博士,副教授,主要研究领域为自然计算、网络与信息安全;王永(1979-),男,硕士研究生,主要研究领域为计算智能;孙星明(1963-),男,博士,教授,博士生导师,主要研究领域为数字水印、网络与信息安全。
收稿日期:2007-11-19 **修回日期:**2008-01-28

$$V_{id}(t+1)=\omega V_{id}(t)+\eta_1\text{rand}()(P_{id}(t)-X_{id}(t))+\eta_2\text{rand}()(P_{gd}(t)-X_{id}(t)) \quad (1)$$

其中 V_{id} 表示第 i 个粒子第 d 维上的速度,ω 为惯性权重,η_1、η_2 为调节 P_{id} 和 P_{gd} 相对重要性的参数,rand()为随机数生成函数。这样,可以得到粒子的下一位置:

$$X_{id}(t+1)=X_{id}(t)+V_{id}(t+1) \quad (2)$$

传统 PSO 的基本算法步骤描述如图 1 所示。

```
Initialize the particle population randomly
Do
  For I=1 to population Size M
    Calculate fitness values of each particle
    Update Pid if the current fitness value is better than Pid
    Determine Pid for each particle
    Choose the Pgd from Pid
    For d=1 to dimension D
        Calculate particle velocity according to (1)
        Update particle position according to (2)
Until termination criterion is met
```

图 1　传统粒子群算法流程

3　面向 TSP 的 N-EPSO 算法

3.1　TSP 的描述

TSP 是运筹学、图论和组合优化中的 NP 难题,常被用来验证智能启发式算法的有效性。TSP 问题描述为:给定 n 个城市及两两城市之间的距离,求一条经过各城市一次且仅一次的最短路线。设 d_{ij} 为城市 i 与 j 之间的距离,即弧(i,j)的长度。引入决策变量 x_{ij},定义如下:

$$x_{ij}=\begin{cases}1 & \text{if visited } j \text{ after visited } i\\ 0 & \text{else}\end{cases} \quad (3)$$

TSP 的目标函数可以表示为:

$$\min Z=\sum_{i,j=1}^{n}x_{ij}d_{ij} \quad (4)$$

TSP 描述非常简单,但随着需要访问城市数目的增加,会出现所谓的“组合爆炸”现象。所以,城市数目比较多的时候使用穷举搜索策略几乎是不可能做到的。

3.2　适应 TSP 的 N-EPSO 编码方案

传统的 PSO 算法虽然成功地应用于连续优化问题中,但在解组合优化问题时,如何针对解空间的特性进行没有冗余信息的编码,是非常关键的工作。下面给出一种顺序表存储的链表结构编码即粒子位置的表示为:

$$X=\{x_1,x_2,\cdots,x_i,\cdots,x_N\},1\leqslant i\leqslant N,1\leqslant x_i\leqslant N \quad (5)$$

式中 N 是城市数,对于第 i 维的数据 x_i,表示一条边(i,x_i),即在访问完城市 i 后将访问城市 x_i,而访问完 x_i 后将依次访问 x_{xi},依此类推,整个访问序列构成了一个循环单链表。例如对 6 个城市,序列(6,4,5,3,1,2)代表的访问顺序为 1->6->2->4->3->5->1。

3.3　协同加法“⊕”的定义

结合 GA 中个体交叉互换信息的思想对 PSO 算法进行改造,引进离散位置的新的加法运算“⊕”,对粒子进行更新操作,对于$(x_1=x_1\oplus x_2)$而言,定义如下:

$$\begin{cases}\phi & \text{if } x_1[i]=x_2[i]\\ x_1[i]\leftrightarrow x_1[x_2[i]];x_1[i]\leftrightarrow x_1[prev(x_2[i])] & \text{else}\end{cases} \quad (6)$$

其中 $prev(x_2[i])$是 $x_2[i]$在 x_1 中的前驱节点,执行算法为:

```
Procedure Add(X1,X2)
Begin
  if X1[i]≠X2[i]
    j=X2[i]
    Swap(X1[i],X1[j])
    j=prev(j)
    Swap(X1[i],X1[j])
End
```

例 x_1={2,5,6,3,4,1},x_2={5,4,6,1,3,2},对于 $x_1[1]\oplus x_2[1]$,因为 $x_2[1]$=5,5 在 x_1 中的前驱为 2。计算过程为:$x_1[1]$与 $x_1[5]$互换;得到 x_1'={4,5,6,3,2,1},然后 $x_1[1]$与 $x_1[2]$互换,得到 x_1={5,4,6,3,2,1}。这里可以看出,每次“⊕”运算后,x_1 中有 3 个元素的位置被改变,即将改变个体 x_1 代表的回路中的三条边;新产生的边中至少有一条边属于 x_2,它是(i,x_{2i});如果不考虑重复改变相同边的情况,当“⊕”操作的概率 P_a 为 1/3 时,将使原个体中所有的边都被改变。

3.4　粒子的更新方程

由以上分析,可以定义粒子的更新方程如下:

$$x_{id}=\alpha(x_{id}\oplus p_{id})+\beta(x_{id}\oplus p_{gd}) \quad (7)$$

其中 α 和 β 为(0,1)之间的随机数,分别代表粒子与历史最好位置和全局最好位置协同相加的概率,且 $\alpha+\beta=1$,当 α 被选中时,β 为 0;β 被选中时,α 为 0。显然,α 越大则代表粒子保留自身成分越多;反之,β 越大则代表群体向最好位置靠拢的趋势越强。一般可以根据城市的数目或者随着进化代数的递增自动调节 α 和 β。执行算法为:

```
Procedure Update(X)
Begin
  For d=1 to dimension D
    if rand(0,1)<Pa   /*Pa 为参加协同加法操作的概率 */
    begin
        if rand(0,1)<α
            Add(Xid,Pid)
      else
            Add(Xid,Pgd)
    end
End
```

3.5　变异算子

算法运算后期群体的多样性可能会有所下降。为了克服这一点,首先粒子中每一维参与“⊕”操作的概率 P_a 不能太高;另外有必要引进一个变异算子来保持种群的多样性,变异算子描述如下:

```
Procedure Mutation(X)
Begin
  For d=1 to dimension D
    if rand(0,1)<Pim   /*Pim 为个体中每一维变异的概率 */
    begin
        随机产生一个合法的数 m
        随机产生一个合法的位置 i
        Add(Xid,m)
    end
End
```

4 N-EPSO 求解 TSP 的实现

算法结构简单,容易实现,描述如图 2 所示。

```
Initialize the particle population randomly
Do
  For I=1 to population Size M
    Calculate fitness values of each particle
    Update Pi if the current fitness value is better than Pi
    Determine Pi for each particle
    Choose the Pg from Pi/*Pg=Min(Pi)*/
    Update(X[i])
    If(rand(0,1)<Pm) /*Pm 为每一个粒子变异的概率 */
        Mutation(X[i])
Until termination criterion is met
```

图 2 N-EPSO 算法流程

这里 *Pm* 与上面 3.5 节中出现的 *Pim* 是不同的,*Pm* 代表群体中变异的个体所占的百分比,而 *Pim* 表示变异个体中的每一维参与变异操作的概率。由以上分析可知,当 *Pim*=0.2 时,在理想情况下(即每次变异操作都是作用于不同的边),该变异个体被改变的边的比例约为 0.2×3=60%。

5 实验分析

文献[4]中采用 14 个点的 TSP 标准问题验证程序的效果,为了方便比较,本文也采用这个典型的 TSP 优化模型,14 点 TSP 的问题描述请参见文献[4]。本文中的实验硬件平台为 PC(Pentium® 4 CPU 3.0 GHz,512 M RAM),采用基于 WinXP OS 的 VC++6.0 作为实验仿真工具,对上述组合优化问题进行进化求解,每个实验均重复进行 30 次。

5.1 实验参数

为了说明参数 α 和 β 对群体多样性和收敛性的影响,先设定 *Pa*=0.3、*Pm*=0.2、*Pim*=0.2,然后分别使 α 和 β 取不同的值对算法进行测试,针对每一对参数重复计算 30 次,在都收敛到最优解 30.878 5 的情况下,得到的平均迭代次数 *avg*、最大迭代次数 max,以及最少迭代次数 min 统计如表 1 所示。

表 1 α 和 β 取不同值时收敛代数统计

α	β	avg	max	min
0.1	0.9	684	9 504	14
0.2	0.8	664	7 407	21
0.3	0.7	672	7 366	17
0.4	0.6	699	9 852	26
0.5	0.5	718	5 774	28
0.6	0.4	804	7 485	56
0.7	0.3	864	6 816	67
0.8	0.2	1 238	8 390	201
0.9	0.1	1 759	12 526	162

数据显示,从大体上来说,随着 α 增大,收敛速度减慢;反之 β 增大,则收敛速度加快;另外当 α 和 β 相差比较大时,出现了局部波动现象,比如当 α=0.1 时的平均收敛速度要慢于 α=0.2 时的平均收敛速度,这是因为当群体中大多都向最优解靠拢时,使得搜索空间大大降低,很容易陷入局部最优,此时起进化作用的是变异算子,从而出现了 α 大也可能使收敛速度更快。

由表 1 中对 α 和 β 的统计分析,以及前面已经讨论的内容,做下面的实验时采用的参数如表 2 所示。

表 2 实验参数

粒子个数	α	β	*Pa*	*Pm*	*Pim*
100	0.5	0.5	0.3	0.2	0.2

5.2 实验结果

实验 1 全局搜索能力比较。

文献[4]中的实验数据说明参考算法收敛到最优解时的平均迭代次数是 20 000,本文用 N-EPSO 算法随机运行 30 次,得出达到该全局最优解的平均迭代次数以及实际迭代收敛机器时间等数据如表 3 所示。

表 3 N-EPSO 算法的实验结果

解空间的大小:14! /(14*2)=3 113 510 400
平均迭代次数:718
最少迭代次数:28
最多迭代次数:5 774
平均机器时间:143 ms
平均搜索空间/解空间:0.002 3%
路径长度:30.878 5(Equal to the optimal solution)

表 4 文献[4]中算法的实验结果

解空间的大小:14! /(14*2)=3 113 510 400
粒子个数:100
平均迭代次数:20 000
平均机器时间:60 796 ms
搜索空间/解空间:0.064%
路径长度:30.878 5(Equal to the optimal solution)

实验 2 搜索效率比较。

为了分析算法的收敛效率,把算法的总迭代次数都规定为 1 000(其他实验参数参见表 2)时,得出如下的实验数据。30 次 N-EPSO 算法实验的平均迭代优化解与文献[4]中算法的数据比较如表 5 所示。

表 5 两种算法的搜索效率比较

算法	迭代次数	平均最优值	收敛概率
N-EPSO	1 000	30.916 8	93.3%
文献[4]中算法	1 000	34.006 7	3.3%

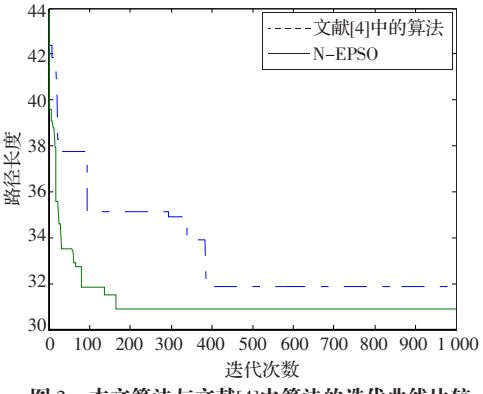

图 3 本文算法与文献[4]中算法的迭代曲线比较

比较可知,本文提出的 N-EPSO 算法搜索效率要比文献[4]中的算法高,在都得到全局最优路径(有时两个解互为逆序)的情况下,平均迭代次数只需要文献[4]中的 3.59%,实际计算机

(下转 75 页)