一种改进并行遗传算法解决 TSP

陈妍峰,田有先
CHEN Yan-feng,TIAN You-xian

重庆邮电大学 计算机科学与技术学院,重庆 400065
College of Computer Science and Technology,Chongqing University of Posts and Telecommunications,Chongqing 400065,China
E-mail:chenyfeng2004@126.com

CHEN Yan-feng,TIAN You-xian.Travelling Salesman Problem based on a reformed parallel genetic algorithm.Computer Engineering and Applications,2008,44(27):62-64.

Abstract: The operation of the genetic algorithm of Travelling Salesman Problem(TSP) needs lots of time and it is easy to fall into the local optimal solution.In order to avoid the problem,the parallel compound genetic algorithm is proposed.The method,which avoids the local optimal problem and reduces the time of the operation,makes use of the parallel of the selection,the cross,the variation.The amount of species distributes the average to the CPU for operating in the environment of MPI.The experience proves the time of the operation less than the simple genetic algorithm and strengthens the ability of the global optimal solution.

Key words: parallel;genetic algorithm;Message Passing Interface(MPI);Travelling Salesman Problem(TSP)

摘 要:针对旅行商问题(Travelling Salesman Problem,TSP)的遗传算法的大规模操作,需要大量运算时间而且容易造成局部最优解,提出一种并行混合遗传算法。该方法基于 MPI 并行环境,利用种群中选择、交叉、变异操作的并行化,将种群中个体平均的分配到处理器中进行操作,有效地避免局部最优解的出现和减少算法的运行时间。实验证明该方法相对于简单遗传算法具有更强全局寻优能力以及耗费更少的操作时间。

关键词:并行;遗传算法;消息传递接口;旅行商问题
DOI:10.3778/j.issn.1002-8331.2008.27.020 **文章编号:**1002-8331(2008)27-0062-03 **文献标识码:**A **中图分类号:**TP301.6

1 引言

旅行商问题(Travelling Salesman Problem,TSP)是一个具有重要理论意义和广泛应用价值的组合优化难题,由于在工业、农业、国防、商业,特别是交通路线等方面的大量运用,因此引起了数学、物理、计算机等诸多领域研究者的关注。由于可能的路径总数与城市数目 N 是成指数增长的。所以一般很难精确地求出其最优解,因此寻求一个有效的近似求解算法是十分必要。

目前求解这个问题主要有模拟退火算法、蚂蚁算法、遗传算法等,其中运用得最为广泛的就是遗传算法。目前求解 TSP 问题的遗传算法主要是混合遗传算法。但是由于混合遗传算法需要引入局部搜索,造成局部最优解,而且增加编码变换操作,造成了运行时间的增加。为了避免以上的缺陷,许多学者作了大量的工作。如文献[1]提出了二进制编码改进遗传算法,文献[2]提出了量子遗传算法,文献[3]提出了离散赌轮选择算法 EPMX 交叉算子和 Dmutation 变异算子,对遗传算法的各个算子进行改进。本文在文献[3]的基础上,进一步改进运行的结构,提出并行混合遗传算法。该算法结合混合遗传算法,基于 MPI 的并行环境,通过各个个体选择、交叉、变异算子[4]的并行化。有效地保证全局最优解和减少算法所消耗时间。

2 问题的描述

TSP 问题的数学模型:设有城市 $V=\{V_1,V_2,\cdots,V_n\}$ 的访问顺序为 $T=\{T_1,T_2,\cdots,T_n\}$,其中 $T_{N+1}=T_1(T_i\in V,i=1\sim N)$,则问题就化为求 $\min\limits_{T\in\alpha}(\sum\limits_{i=1}^{N}d_{i,j+1})$,其中 α 是表示经过城市走一次且仅一次,再回到原点的所有不重复回路。

本文解决 TSP 问题用的编码是最简单、最直观的方法。首先是对各个不同的目标对象编写序号,即用 0 表示目标城市,1~N 表示 N 座目标城市。为了使得很好的计算出最小路径,定义了二维数组(I,J),其中 I 表示目标城市的序号,J 表示各个目标城市之间的权值,即各城市间的距离。由此得出来适应度函数:

$$F(x)=\begin{cases}C_{\max}-f(x) & \text{if } f(x)<C_{\max}\\ f(x) & \text{if } f(x)\geqslant C_{\max}\end{cases} \quad (1)$$

其中 $f(x)$ 是属于目标函数,表示经过了每个城市一次且仅一次的,最后回到开始城市的路程的总和。记为 $\sum\limits_{i=1}^{N}d_{i,j+1}+d_{i+1,1}$。其中 $C_{\max}$ 用来控制收敛的速度。$C_{\max}$ 取值越大将会使得收敛得速度越快,为了保证初期的多样性,要延缓收敛速度,尽量避免早熟的情况的发生。因此在算法的迭代初期使用不适应度函数:

基金项目:重庆市科委基金项目(No.CST2005BB0061)。
作者简介:陈妍峰(1984-),男,硕士研究生,主要研究领域:并行计算、遗传算法。
收稿日期:2007-11-14 **修回日期:**2008-02-25

$$F(x)=-f(x)=-\sum_{i=1}^{N} d_{i,j+1}+d_{i+1,1} \quad (2)$$

在迭代的后期使用式(1)适应度函数。这样就可以保证迭代过程多样性和收敛的速度较快。

本文采用适应度函数比例选择，设定选择算子为：

$$P\{x_i\}=\frac{\sum_{k=1}^{N} F\{x_k\}-F\{x_i\}}{(N-1)\sum_{k=1}^{N} F(x_k)} \quad (3)$$

通过计算出选择算子后，再对种群进行轮盘赌选择。交叉算子本文采用扩展部分匹配交叉(EPMX)[3]，有效解决匹配的过程中，部分基因相同，部分基因不同的情况出现。变异算子本文采用 Dmutation 算子。利用变异算子调整整个编码中的个别基因值，提高局部优化值逼近于全局最优解，达到全局搜索能力同时可以防止种群中的个体早熟的情况。

3　改进并行遗传算法

3.1　并行遗传算法的原理

并行遗传算法(Parallel Genetic Algorithm，PGA)是一种适应度于复杂约束优化问题的全局优化能力。并行遗传算法主要有 3 类[6]：(1)主从式并行遗传算法(如图 1 所示)；(2)粗粒度并行遗传算法；(3)细粒度并行遗传算法。其中主从式并行遗传算法是主处理器监控整个染色体种群，并执行选择操作；各个从处理器接受来自主处理器的个体进行重组交叉和变异，产生新一代个体，但是在适应度评价很费时且远远超过通信时间的情况下才有效，否则通信时间超过计算时间，反而会降低速度。粗粒度模型将种群分成若干个子群并分配给对应的处理器，每个处理器不仅独立计算适应度，而且独立进行选择，重组交叉和变异操作，还要定期地相互传送适应度最好的个体，从而加快满足终止条件的要求。但是相互间的通信产生大量的时间，降低算法的效率。

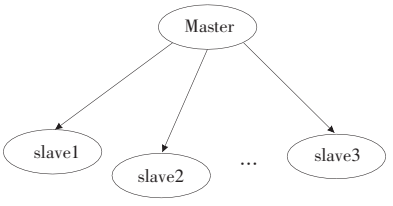

图 1　主从式并行遗传算法图

3.2　改进的并行遗传算法

为了避免上面所说的缺点，本文将主从式模型和粗粒度模型相结合，提出一个新的并行遗传算法模型，监控整个种群，将个体平均的分配到从处理器中进行选择、交叉、变异操作，定期返回适应度最好的个体，选择其中的最佳个体替换各个从主处理器中的局部最优解，产生新一代个体。程序流程图如图 2 所示。

并行程序的核心是任务分配，如何将任务均匀的分配到各处理器是影响并行程序性能的一个主要因素。本文采用块分配的方法。所谓块分配，是指将种群数目连续的分成若干个任务块，每个处理器负责一个块的计算。种群的分配与收集是通过 MPI 的组通信函数实现的，由于种群数目可能不被处理器个数整除，因此发送到个处理器的个体的个数可能不相同，因此使用组通信函数 MPI_Scatter 和 MPI_Gather 实现种群的分配与收集，其中用 MPI_Barrier 实现种群中所求值的一致性。

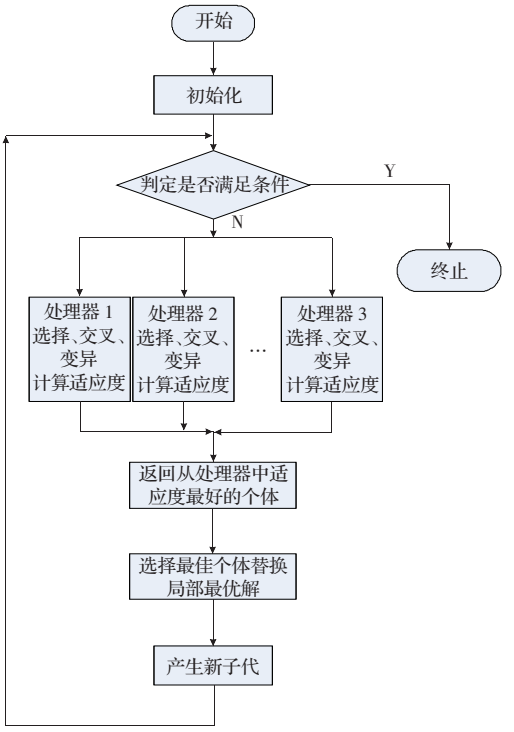

图 2　程序流程图

4　并行性分析

4.1　串行执行时间

首先讨论算法的串行执行时间。算法时间主要包括 3 个部分：第 1 部分是初始化种群时间；第 2 部分是选择、交叉、变异算子运行时间；第 3 部分是计算适应度时间。

设城市的数目为 C，种群数目为 n_1，开始设随机获取交叉位置 r_1，初始化种群的时间复杂度为 $L(n_1)$，由于每个个体都要取值 C 次，则获取 n_1 个个体时间复杂度为：$L(n_1)=C*n_1$。

另设选择、交叉、变异算子时间复杂度为 $L'(n_1)$，通过式(3)计算选择算子，再通过轮盘赌选择个体，时间复杂度即 $C*n_1+n_1$；通过 EPMX 完成交叉操作，其中包括了随机获取交叉个体的时间 P_c*n_1，和运行交叉算子的时间复杂度 $(r_1*r_1+C)*P_c*n_1$，则运行时间复杂度为：$(r_1*r_1+C+1)*P_c*n_1$；通过变异算子的时间包括了随机获取变异时间和变异算子运行的时间复杂度 $2*p_v*n_1$；因此可得：$L'(n_1)=2*p_v*n_1+(r_1*r_1+C+1)*P_c*n_1+C*n_1+n_1$，最后适应度计算时间复杂度为 $L''(n_1)=C*n_1$。由上可得串行执行时间复杂度为：$L'''(n_1)=2*p_v*n_1+(r_1*r_1+C+1)*P_c*n_1+3C*n_1+n_1$。

4.2　并行性分析

由 K 台处理器运行的种群数为 N 遗传算法。通过分块处理种群，从处理器处理时间复杂度为：$L^4(n_1)=2*p_v*n_1/k+(r_1*r_1+C+1)*P_c*n_1/k+3C*n_1/k+n_1/k$，即 $L^4(n_1)=L'''(n_1)/K$，设通信时间为 T_c，K 台处理器规约的时间复杂度为 $S(n_1)$。并行执行时间为：。由串行执行时间 T_s 和并行执行时间 T_p 得到加速比为：$\frac{T_s}{T_p}=\frac{L'''(n_1)}{L^4(n_1)+T_c+S(n_1)}$ 只有 n_1 值越小时，通信时间才会减小，规约的时间减小，加速比趋近于 K。但是当 n_1 太小时，通信时间大于算法的执行时间，同样会影响加速比。

5 实验与分析

给定 12 个城市的权值,举证如下:

$$W=\begin{pmatrix}0&50&2&95&86&14&78&95&20&12&23&52\\50&0&44&51&67&57&74&51&52&21&74&24\\2&44&0&95&42&12&24&74&62&35&45&12\\95&51&95&0&5&19&86&74&92&51&27&14\\86&67&42&5&0&59&5&100&17&34&46&47\\14&57&12&19&59&0&80&32&89&25&23&41\\78&74&24&86&5&80&0&32&89&31&35&61\\95&51&74&74&100&99&32&0&11&51&22&13\\20&52&62&92&17&34&89&11&0&76&86&19\\12&21&35&51&34&25&31&51&76&0&85&25\\23&74&45&27&46&23&35&22&86&85&0&43\\52&24&12&14&47&41&61&13&19&25&43&0\end{pmatrix}$$

根据上述的算法的求得。例中 12 个城市所组成的模型,总共有(12-1)! /2=19 958 400 条路径可以供选择。

通过对种群数目 1 200、4 800、8 400、9 600 的遗传算法求解。比较算法的加速比和效率。取交叉比例 P_c=0.25,变异比例 P_v=0.01, 实验环境:Windows XP 操作系统,VC 2005,MPICH2 并行环境。如表 1 所示。

从表 1 可以看出,对于相同的处理器数目,随着种群数目的增加,局部最优解就越接近全局最优解。

对于种群数目 N=1 200,4 800 的遗传算法的求解,加速比都接近于 K,这与前面的并行性分析基本一致。当种群数目 N=8 400,9 600 时,加速比小于当种群数目 N=1 200,4 800 情况下得到的加速比, 这是由于通信时间随着种群数目的增加而增加,造成了效率低。

表 1 本文算法的加速比及效率

种群数目 N		1 200	4 800	8 400	9 600
处理器数目 K=1	求得最优解	261	243	219	212
	时间	21.48	113.21	249.13	303.34
处理器数目 K=2	时间	12.49	59.90	156.69	204.96
	加速比	1.72	1.89	1.59	1.48
	效率	0.860	0.945	0.795	0.740
处理器数目 K=3	时间	8.84	40.15	112.22	153.20
	加速比	2.43	2.82	2.22	1.98
	效率	0.81	0.94	0.74	0.66
处理器数目 K=4	时间	6.22	30.51	81.42	109.51
	加速比	3.45	3.71	3.06	2.77
	效率	0.863	0.928	0.765	0.693

6 结论

本文讨论的是基于 MPI 的并行遗传算法用于解决旅行商问题。通过并行计算扩大种群数目使得局部搜索更加收敛于全局最优解,并且有效地提高了算法的运行时间,是一个有效算法。

参考文献:

[1] 傅玉芳.基于遗传算法求解 TSP 问题的一种新方法[J].计算机应用研究,2004(9):45-49.

[2] 王宇平,李英华.求解 TSP 的量子遗传算法[J].计算机学报,2007(30):748-754.

[3] 周涛.基于改进遗传算法的 TSP 问题研究[J].微电子学与计算机,2006.

[4] 王小平,曹立明.遗传算法——理论、应用与软件实现[M].西安:西安交通大学出版社,2000.

[5] 喻菡.遗传算法求解 TSP 的研究[D].成都:西南交通大学,2006.

[6] 陈国良.并行算法实践[M].北京:高等教育出版社,2004.

(上接 48 页)

当 $state_1$ 从当前状态变迁到下一个状态时, 由多项式组乘积的定义,乘积 $F_{s1}=F_1F_2F_3F_4F_5F_6F_7F_8$ 表示点$\langle x_1,\cdots,x_5,x_1'\cdots x_5'\rangle$对应变量的适当赋值。同样对 $state_2$ 和 $turn$,可得 F_{s2} 和 F_{turn}。令 $Z=\{x_1(x_1+1),\cdots,x_5(x_5+1),x_1'(x_1'+1),\cdots x_5'(x_5'+1)\}$,则系统的变迁关系可以表示为:$T=F_{s1}\cup F_{s2}\cup F_{turn}\cup Z$。

得到了变迁关系的多项式后,就可以利用吴方法检验系统的性质了。待检验的性质是:$EF\ (state_1=c_1\wedge state_2=c_2)$, 令 $\phi=(state_1=c_1\wedge state_2=c_2)$对应多项式组为:$P=\{x_1(x_1+1),x_2+1,x_3,x_4+1,x_5\}$。找 $\lambda y.\phi\vee EXy$ 的最小不动点,即 $\mu y.(\phi\vee EXy)$。

为了找到 EX_y 对应的零点,其中 F_y 代表 y。满足映射:$f:x_i\mapsto x_i'(1\leqslant i\leqslant 5)$,计算特征列 $T\cup f(F_y)$用 G_y 表示。($G_y\cap K[x_1,\cdots,x_5]$)$P$ 对应公式 $\phi\vee EXy$,在这种情况下,计算 $\lambda y.\phi\vee EXy$ 的最小不动点。

用 Maple 软件在奔腾Ⅳ,CPU 2 GHz,480 MB 内存上运行该例子。令 G 是对应 $\lambda y.\phi\vee EXy$ 的最小不动点的多项式组,则 $G=\{x_1(x_1+1),x_2+1,x_3,x_4+1,x_5\}$,初始状态由多项式组表示:$F_{init}=\{x_1(x_1+1),x_2,x_3,x_4,x_5\}$。存在一个 F_{init} 的零点不满足 G,因此得到刻画 EF_ϕ 是错误的。应用吴方法只花费了 6.7 s,901 KB 内存。在同样的机器上 SMV 花费 0.1 s,980 KB 内存。采用吴方法进行模型检验所用内存较少,算法还没有优化,从长远的观点来看,吴方法对解决状态爆炸问题是有效的。

5 结束语

在符号模型检验中利用 BDD 技术的方法能缓和状态组合爆炸问题,但并没有彻底解决问题。本文给出了应用吴方法来进行符号模型检验的方法。系统的初始状态、变迁关系和表示系统性质的 CTL 公式都用多项式来表示, 然后用吴方法计算多项式组的特征列, 从而检验系统是否具有用 CTL 公式描述的性质。给出的案例表明了这种新方法能有效地缓和内存爆炸问题。在实际应用中还可以优化寻找特征列的算法。

参考文献:

[1] McMillan K L.The SMV system[EB/OL].(2000).http://www-2.cs cmu.edu/~modelcheck/smvmanual.ps.

[2] Brayton R K.VIS:a system for verification and synthesis[C]//Proc 8th International Conference on Computer-Aided Verification (CAV'96).[S.l.]:Springer-Verlag,1996:428-432.

[3] Cimatti A.NuSMV 2:an OpenSource tool for symbolic model checking[C]//Int'l Conf on Computer-Aided Verification(CAV 2002).[S.l.]:Springer-Verlag,2002:27-31.

[4] Ritt J F.Differential algebra[M].New York:Amer Math Sco,1950.

[5] Wu W T.Basic Principles of mechanical theorem proving in geometries[J].J of Automated Reasoning,1986,2(4),221-252.

[6] Mao W B,Wu J Z.Application of Wu's method to symbolic model checking[C]//ISSAC'05.[S.l.]:ACM Press,2005.