几种仿生优化算法的比较研究

熊伟平,曾碧卿
(华南师范大学 计算机学院,广东 广州 510631)

摘 要:仿生优化算法是一类模拟自然生物进化或者群体社会行为的随机搜索方法的统称。由于这些算法求解时不依赖于梯度信息,故其应用范围较广,特别适用于传统方法难以解决的大规模复杂优化问题。阐述了三种典型的仿生优化算法——遗传算法、蚁群算法和混合蛙跳算法各自的产生背景、基本思想以及实现步骤,然后深入分析讨论了它们的异同之处与适用范围,最后指出了仿生优化算法今后的发展趋势和研究方向,其中提出的一些改进思路对进一步的研究工作有一定的理论意义和应用价值。

关键词:仿生优化;遗传算法;蚁群算法;混合蛙跳算法
中图分类号:TP301.6 **文献标识码**:A **文章编号**:1673-629X(2010)03-0009-04

Studies on Some Bionic Optimization Algorithms

XIONG Wei-ping, ZENG Bi-qing
(School of Computer Science, South China Normal University, Guangzhou 510631, China)

Abstract: Bionic optimization algorithms are stochastic search methods that mimic the metaphor of natural biological evolution or the social behavior of species. They are widely used independent of gradient, so they are suitable to solve large scale complicated optimization problems which are hard to be solved in traditional methods. First expatiates basic ideas and process realization of three classic bionic optimization algorithms: genetic algorithm, ant colony optimization, and shuffled frog-leaping algorithm. Then their similarities and differences are discussed. Finally, future research directions are pointed out . Some improved ideas have theoretical significance and practical value as to relative works.

Key words: bionic optimization; genetic algorithm; ant colony optimization; shuffled frog-leaping algorithm

0 引 言

在计算机科学、自动化、管理和工程技术等领域里人们经常会碰到组合优化问题,其中的很多问题如旅行商问题(TSP),指派问题(QAP),车间作业调度(JSP)等都已被证明是NP完全问题。用传统的单纯形法或非线性规划这些基于数学的优化方法解决此类问题时,计算时间随着问题规模的增大呈指数级延长,且要求目标函数有较严格的数学特性,这大大限制了其应用的范围。为了克服这些困难,科学家们从生物系统的进化和自适应性现象得到灵感,提出了一些以搜索近优解为目标的仿生优化算法。

仿生优化算法是一类模拟自然生物进化或者群体社会行为的随机搜索方法的统称。自从20世纪70年代产生第一个仿生优化算法——遗传算法以来,越来越多的研究者投身其中,又先后提出了蚁群算法、微粒群算法、捕食搜索算法、人工鱼群算法、混合蛙跳算法等一系列仿生算法。由于这些算法求解时不依赖于梯度信息,其应用范围较广,在许多传统方法难以解决的大规模复杂问题中都已体现出了优异的性能。因此,它们的出现为NP完全的组合优化问题求解提供了一条全新的途径,并作为新兴的演化计算方法已越来越受到国内外研究者的关注[1]。

1 几种算法的基本思想

1.1 遗传算法

遗传算法(Genetic Algorithm, GA)[2]是由美国密歇根大学的John Holland教授及其学生于20世纪60年代末到70年代初提出的。在1975年出版的《Adaptation in Natural and Artificial Systems》一书中,Holland系统地阐述了遗传算法的基本理论和方法。后来De Jong和Goldberg等人做了大量的工作,使遗传算法更加完善。由于遗传算法求解复杂优化问题的巨大潜力

收稿日期:2009-07-20;**修回日期**:2009-10-19
基金项目:广东省自然科学基金项目(8151063101000040)
作者简介:熊伟平(1983-),男,江西赣州人,硕士研究生,研究方向为分布式计算、智能优化算法;曾碧卿,博士,副教授,研究方向为分布式处理、P2P计算、并行I/O。

及其在工业工程、人工智能、生物工程、自动控制等各个领域的成功应用,该算法得到了广泛的关注[3]。

遗传算法是根据自然进化论与遗传变异理论为基础求解全局最优解的仿生型算法,其本质是一种求解问题的高效并行全局搜索算法。该算法将问题的求解表示成染色体(Chromosome),从而构成种群,再将它们置于问题的环境中,根据适者生存的原则,从中选择出适应环境的染色体进行复制后,通过交叉(Crossover)、变异(Mutation)产生出新一代更适应环境的染色体群,这样一代代地不断进化,最后收敛到一个最适合环境的个体上,求得问题的最优解。

基本的遗传算法包括 4 个要素:编码机制、适应度函数、选择策略、遗传算子。下面对其分别作简要说明。

(1) 编码机制。

用遗传算法求解之前,首先要对问题进行建模,将解空间进行编码,把每个解编码成字符串的形式,称为染色体。整个算法就是对染色体或其部分实施各种操作。最常用的编码机制有二进制编码和实数编码。

(2) 适应度函数。

适应度函数用来评估种群中每个个体对其生存环境的适应能力。它是进化过程中个体选择的重要依据,将直接影响遗传算法的收敛性和解的质量。适应度函数基本上依据优化的目标函数来确定,且要求适应值为非负,适应值越大表示其对环境的适应能力越强。

(3) 选择策略。

借鉴自然进化理论中适者生存的思想,在每一代群体里使适应度高的个体有更大的概率被选择用来繁殖后代,保证优良基因遗传给下一代个体。通常采用按比例的适应度分配方法,即若种群规模为 M,个体 i 的适应值为 f_i,则该个体被选择的概率即为 $P_i = f_i / \sum_{i=1}^{M} f_i$。得到选择概率后,用轮盘赌法来实现选择操作。

(4) 遗传算子。

遗传算子包括交叉和变异,目的是模拟每一代中创造后代的繁殖过程。交叉是按照一定的交叉概率在种群中随机选择两个个体,交换它们的部分基因从而产生两个新个体的过程。交叉率一般取较大值,典型地在 0.6 至 0.9 之间。交叉的意义在于产生新的基因组合,而不是一代代重复同样的个体。变异是以很小的变异概率(一般小于 0.1)在染色体上自发地产生随机的变化,如替换一个或多个基因。变异的作用是保持种群的多样性,能够有效地跳出局部最优解。

传统遗传算法的基本流程可描述为:

Step1 初始化。随机地产生 M 个解,作为初始种群。

Step2 根据设计的适应度函数,计算种群中每个个体的适应值。如果算法收敛或者达到最大迭代代数,则输出最佳个体及最优解,结束计算。否则转 Step3。

Step3 按照适应值和选择策略,选择出用于繁殖下一代的个体,适应值高的个体被选中的概率也高。

Step4 按照指定的交叉率和交叉方法,对种群执行交叉运算。

Step5 按照指定的变异率和变异方法,对种群执行变异运算。

Step6 经过选择、交叉、变异后,产生了下一代的种群,转 Step2。

1.2 蚁群算法

蚁群算法(Ant Colony Optimization, ACO),又称蚂蚁算法,是意大利学者 Dorigo M 等人在 1991 年提出的一种模拟蚂蚁群体觅食行为方式的仿生优化算法[4]。当蚂蚁在寻找食物时,每个走动的蚂蚁都会在其经过的路径上分泌一种叫做信息素(Pheromone)的化学物质,而且能感知这种物质的存在及其浓度。每条路径上信息素的数量,会反映出其它蚂蚁选择该路径的概率,蚂蚁趋向于朝着信息素浓度高的方向移动。在较短路径上的信息素会很快地增加,使得最终所有的蚂蚁将选择最短的路径。目前蚁群算法已经渗透到各个应用领域,从一维静态优化问题到多维动态优化问题,从离散问题到连续问题,解决了许多复杂优化和经典 NP-C 问题[5]。

蚁群算法最初被用于解决经典的对称旅行商问题(STSP),取得了满意的效果。人工蚂蚁在城市之间按照一定的规则转移,遍历完所有的城市即得到一个解。具体的执行过程为:在 n 个城市随机地放置 m 只蚂蚁,记录下出发点,开始时所有路径上的信息素数量相同。每只蚂蚁根据路径上的保留信息独立地选择下一个城市。在 t 时刻蚂蚁 k 从城市 i 按照下面的概率公式转移到城市 j:

$$P_{ij}^{k}(j) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{k \in \text{allowed}_k}[\tau_{ik}(t)]^{\alpha}[\eta_{ik}]^{\beta}}, & \text{若 } j \in \text{allowed}_k \\ 0, & \text{否则} \end{cases} \tag{1}$$

其中,$\tau_{ij}(t)$ 表示 t 时刻城市 i 和 j 之间的信息素数量;η_{ij} 是路径的启发信息,表示城市 i 转移到 j 的期望程度,通常取为城市 i 和 j 之间的距离 d_{ij} 的倒数,即 $\eta_{ij} = 1/d_{ij}$;α 是信息启发式因子,β 是期望启发式因子,

分别表示信息素和启发函数的相对重要程度;$\text{allowed}_k = \{C - \text{tabu}_k\}$ 表示在 t 时刻蚂蚁 k 下一步允许选择的城市。

当蚂蚁完成对所有 n 个城市的遍历,即一次迭代后对路径上残留的信息素进行更新。在 $(t+n)$ 时刻城市 i 和 j 之间的信息素数量根据公式(2)和(3)的全局调整规则作更新:

$$\tau_{ij}(t+n) = (1-\rho)\cdot\tau_{ij}(t) + \Delta\tau_{ij} \tag{2}$$

$$\Delta\tau_{ij} = \sum_{k=1}^{m}\Delta\tau_{ij}^{k} \tag{3}$$

其中,ρ 表示信息素的挥发系数,作用是防止信息素的无限积累,取值范围为[0,1),而 $1-\rho$ 表示信息的残留系数。$\Delta\tau_{ij}$ 表示本次迭代中路径 (i,j) 上的信息素增量,$\Delta\tau_{ij}^{k}$ 表示第 k 只蚂蚁在本次迭代中留在路径 (i,j) 上的信息素。

基本蚁群算法的实现步骤如下:

Step1 初始化。迭代次数 Nc=0,令所有路径 (i,j) 的初始化信息量 $\tau_{ij}(0) = c$(c 为常数),$\Delta\tau_{ij} = 0$。把 m 只蚂蚁随机地放在 n 个城市上。

Step2 迭代次数 Nc = Nc + 1。

Step3 对每只蚂蚁 k,令其禁忌表 tabu_k 的索引号 $s = 1$,并将其出发城市放入 tabu_k 中。

Step4 禁忌表索引号 $s = s + 1$。

Step5 按照概率转移公式(1),选择蚂蚁将要行走的下一个城市 j,把蚂蚁移动到该城市,并把该城市记入到禁忌表中。

Step6 若没有遍历完所有城市,即 $s < n$,转到 Step4;否则,转到 Step7。

Step7 按照公式(2)和(3)的全局调整规则更新每条路径上的信息量。

Step8 若算法收敛或达到最大迭代代数,则算法结束并输出最优解;否则清空禁忌表并转到 Step2。

1.3 混合蛙跳算法

混合蛙跳算法(shuffled frog - leaping algorithm, SFLA)是一种受自然生物模仿启示而产生的基于群体的协同搜索方法,由 Eusuff M 和 Lansey K 在 2003 年首先提出,并用来成功解决了水资源网络分配问题[6]。这种算法模拟青蛙群体寻找食物时,按族群分类进行思想传递的过程。

在混合蛙跳算法中,种群由许多相同结构的青蛙组成,每只青蛙代表一个解。整个种群被分为多个子群,不同的子群可看成是具有不同思想的青蛙集合。每个子群执行局部搜索,子群内的每只青蛙有自己的思想,同时会受到其它青蛙的影响,随着子群的进化而进化。当子群进化达到设定的代数后,各个子群之间进行信息传递实现混合运算。这样局部搜索和混合过程交替进行直到满足停止准则。

对于 D 维问题,一只青蛙 i 可以表示为 $F_i = (f_{i1}, f_{i2}, \cdots, f_{iD})$。算法首先随机产生 S 只青蛙作为初始种群,按照每只青蛙的适应度进行降序排序。然后整个种群被划分为 m 个子群,每个子群包含 n 只青蛙,即满足 $S = m * n$。其中,第1只青蛙被放在第1个子群,第2只青蛙被放在第2个子群,一直到第 m 只青蛙被放在第 m 个子群。接着,第 $m+1$ 只青蛙又被放在第1个子群,依此类推,将所有青蛙分配完毕。

在每个子群中,具有最好适应度和最差适应度的青蛙分别记为 F_b 和 F_w。而整个种群中具有最好适应度的青蛙记为 F_g。在子群进化的过程中,每次迭代只更新最差青蛙 F_w 的位置。

调整公式为:

$$C_i = \text{rand}() \times (F_b - F_w) \tag{4}$$

$$F_w = F_w + C_i;(-C_{\max} \leqslant C_i \leqslant C_{\max}) \tag{5}$$

式(4)用于计算更新的步长 C_i,rand() 为0到1之间的随机数;式(5)对 F_w 执行更新,修改青蛙的位置,$C_{\max}$ 为最大步长。如果得到更好的解,则用其替代最差个体;否则用 F_g 代替式(4)中的 F_b,重新计算新解。若仍然得不到更优解,则随机产生一个新解去替换最差个体。重复直到预定的迭代次数,就完成一轮各子群的局部搜索。然后将所有子群的青蛙重新混合排序,划分子群,进行下一轮的局部搜索。如此反复直到满足终止条件。

混合蛙跳算法的实现过程可描述如下:

Step1 随机产生 S 个解(青蛙)作为初始种群。

Step2 计算每个个体的适应度,将种群的所有个体按照适应度降序排序,并记下最好的青蛙个体 F_g。

Step3 把 S 只青蛙分为 m 个子群,每个子群含有 n 只青蛙。

Step4 对于每个子群,找出最好个体 F_b 和最差个体 F_w,然后按照公式(4)和(5)更新最差个体。重复该步骤直到预定的次数。

Step5 混合所有的子群,重新形成一个含 S 只青蛙的完整种群。

Step6 如果满足停止准则,则输出最优解,算法结束;否则转到 Step2。

2 三种算法的比较分析

遗传算法与蚁群算法、混合蛙跳算法都属于模拟自然界生物系统行为或过程的仿生智能算法,主要目标都是求解科学和工程中遇到的各种最优化问题。因

此,它们有很多的相同点:

(1)优化流程遵循相同的结构框架。算法从一组初始解出发,通过适应度函数评估后,在参数的控制下按优化策略更新当前状态,进化得到一组新解,重复以上搜索过程,直至算法终止准则满足,最终得到问题的优化解。编码,适应度函数、状态更新方式、算法参数设置、收敛准则是算法的核心。

(2)算法同时对种群中的各个个体进行搜索寻优,各个体通过直接或间接的通信协同工作,所以寻优过程实际上是种群的进化过程,具有本质的并行性。

(3)都是基于概率的随机搜索算法。算法的步骤均含有随机因素,事件的发生与否带有很大的不确定性,使得其有更多的机会获得全局最优解,比较灵活。

(4)都有很强的鲁棒性。这是因为仿生优化算法不依赖问题本身的严格数学性质,对目标函数和约束函数的要求比较宽松,且当求解一个新问题时,只要设计好适应度函数,算法的其它部分不需作太多修改,所以在不同环境下适用性非常好。

(5)基本思想都来自对某种自然规律的模仿,种群能够通过自组织和自学习提高其适应性,具有人工智能的特点。

(6)不以达到精确最优解为目标,而是更看重计算的速度和效率。算法的理论基础相对比较薄弱,多数成果来自于实验结论。

虽然上述算法都属于仿生优化算法,有很多相似的地方,但由于各种算法提出的初衷和应用背景不同,它们在算法思想和实现手段等方面也存在一些不同之处。

遗传算法将对参数编码后的染色体作为运算对象,而不是参数本身,因此不受函数约束条件的限制。在优化过程中借鉴生物学中染色体和基因等概念,模拟自然界中生物的遗传和进化等机理,应用遗传操作,能够较好地求解无数值概念或很难有数值概念的优化问题。GA 的主要缺点是编码机制的选择对遗传算子的设计影响很大;对于结构复杂的组合优化问题,搜索空间大,搜索时间比较长,往往会出现早熟收敛的情况;对初始种群很敏感,初始种群的选择常常直接影响解的质量和算法效率。

蚁群算法中蚂蚁通过信息素进行间接的通信,随着个体的增加系统通信开销增加的较小,具有很强的发现较好解的能力。引入了正反馈并行机制,加快了进化过程,不易陷入局部最优,具有较强的鲁棒性、优良的分布式机制、易于与其他方法结合等优点。但是蚁群算法每次解构造过程的计算量较大,使得搜索时间很长,且容易出现停滞现象;算法收敛性能对初始化参数的取值比较敏感。

混合蛙跳算法本质上是结合了基于遗传特性的模因算法和基于社会行为的微粒群算法两者的优点[7],达到了全局探索能力和局部开发能力的平衡,有较强的鲁棒性。算法简单易于理解,且只要做细微修改就可适用于不同的场合。缺点是该算法主要适用于连续空间域的优化问题,收敛速度较慢。由于其提出时间不长,到目前为止研究成果很少,应用范围也较窄。

3 结束语

上面阐述和比较了三种典型的仿生优化算法,它们已经在解决不同环境下的优化问题中取得了很好的效果,在其它某些新兴领域也体现出巨大的潜力,但都还存在一些需要进一步研究的问题:

(1)对仿生优化算法的数学理论研究亟待加强,如算法的收敛性证明,鲁棒性分析等。另外尤其重要的是,每种仿生算法都有几个参数,它们的取值对收敛性和解的质量有很大影响,而现在的应用对其都是通过反复实验来确定。如果能够提出合理有效的数学方法来确定参数值,无疑会大大促进仿生优化算法的发展。

(2)仿生算法给出的只是一个基本的计算思想和步骤,因此改进算法步骤以获得更好的计算性能有很大的创新空间。比如,为遗传算法设计新的遗传算子,考虑精英策略和局部搜索机制;修改蚁群算法的信息素更新规则,改善其全局收敛性;在混合蛙跳算法中,加入惯性因子以提高收敛速度,等等。

(3)将几种仿生算法结合起来或者将仿生算法和其它的启发式算法结合产生新的混合优化算法,目前对蚁群算法和遗传算法的融合已有一些成果[8],但是对混合蛙跳算法与其它算法结合的研究较少,还需要不断地发展和完善。

参考文献:

[1] 汪定伟,王俊伟,王洪峰,等.智能优化方法[M].北京:高等教育出版社,2007.

[2] 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,2002.

[3] 朱文龙,丁华福.遗传算法在多目标柔性 Job-Shop 调度中应用[J].计算机技术与发展,2009,19(4):217-223.

[4] Dorigo M,Maniezzo V,Colorni A. Ant System: Optimization by a Colony of Cooperating Agents[J]. IEEE Transactions on Systems,Man,and Cybernetics,1996,26(1):1-13.

[5] 段海滨.蚁群算法原理及其应用[M].北京:科学出版社,2006.

[6] Eusuffm M, Lansey K. Optimization of Water Distribution

(下转第 16 页)

行 9770 的 lite_version:no 改为 yes;

行 9775 的 trace_message:yes 改为 no;

行 9778 的 move_window_by_mouse:yes 改为 no;

行 9827 的 build_vbf_support:yes 改为 no;

行 9834 的 build_type1_support:yes 改为 no;

行 9878 的 build_ime_gb2312_pinyin:yes 改为 no;

行 9912 的 enable_video_qvfb;yes 改为 no;

使用交叉编译器 configure,运行时检查 -lttf、-ljpeg、-lpng 和 -lm 四项都是 yes,证明之前准备都已妥当,使用 make 和 make install 安装 libminigui V1.3.3。

4.3.2 minigui-res-1.3.3 的安装

minigui-res-1.3.3 包含了 MiniGUI 运行时的一些资源文件,包括图标、光标等等,解压后将它放到交叉编译器的资源目录中即可。

4.3.3 mde-1.3.0 的交叉编译

mde-1.3.0 是基于 MiniGUI V1.3.3 的一套示例程序,可以用来检查 MiniGUI 是否安装成功,同时也可以作为开发嵌入式 MiniGUI 程序的参考[3]。进入 mde-1.3.0 目录,使用交叉编译器 configure,同时加上参数 host=arm-linux。之后通过 make 来生成适用于嵌入式 Linux 的二进制文件。

安装完成后,进入 mginit 目录,mginit 已经编译,证明针对嵌入式 Linux 的 MiniGUI V1.3.3 交叉编译环境搭建完成。

4.4 MiniGUI V1.3.3 的移植

搭建 MiniGUI V1.3.3 交叉编译环境是为在 EBD9261 开发板上运行 MiniGUI 程序,因此需要把一些与 MiniGUI 相关的依赖库移植到开发板的文件系统中,使开发板成为一个支持 MiniGUI 的容器。

因为 EBD9261 开发套件中已经配备了嵌入式 Linux 的文件系统,只需对文件系统做一定的修改即可。将文件系统镜像 ramdisk9261.gz 解压为 ramdisk9261。然后将 ramdisk9261 挂载至 ramdisk 目录。之后就可以修改文件系统。将交叉编译器目录下 lib 目录中的相应动态库和静态库复制到/ramdisk/usr/local/lib/中,同时需要把 MiniGUI 运行的资源库复制到相应的目录下。修改 MiniGUI.cfg,使 MiniGUI 适应嵌入式开发板的硬件环境。

为了验证开发板是否已经具备运行 MiniGUI 的库和资源文件,把 mde-1.3.0 中的程序复制到 ramdisk 目录下的 mde 目录中。准备完毕后,卸载 ramdisk9261,并将其压缩为镜像 ramdisk9261.gz。通过 minicom 将文件系统镜像文件烧写到 EBD9261 开发板中,整个工作就做完了。

运行开发板进入系统后,使用终端运行 mginit,可以看到类似 Windows 3.1 的图形界面。到这里,MiniGUI V1.3.3 的交叉编译环境和嵌入式运行环境成功的搭建完成。

5 结束语

随着嵌入式产品应用领域的日益增长,开发出优秀的人机交互界面,是嵌入式发展的趋势,拥有广阔的市场前景。MiniGUI 可以稳定可靠的运行于 Linux 系统,通过上述具体的移植和后续的 MiniGUI 嵌入式软件的开发过程,能快速构建一个嵌入式可视化软件系统,相信这种嵌入式系统将会得到越来越多的应用。

文中创新点:成功实现了图形用户界面 MiniGUI V1.3.3 的移植开发和对开发板 EBD9261 的良好支持,同时也适用于其他多种开发环境,完成了构建嵌入式图形界面系统的前期工作。

参考文献:

[1] 北京飞漫软件. MiniGUI 用户手册 V1.3[EB/OL]. 2003. http://www.minigui.com.

[2] 北京飞漫软件. MiniGUI 编程手册 V1.3[EB/OL]. 2003. http://www.minigui.com.

[3] 周立功. ARM 嵌入式 MiniGUI 初步与应用开发范例[M]. 北京:北京航空航天大学出版社,2006.

[4] 深圳市英贝德科技有限公司. EBD9261 硬件手册[EB/OL]. 2006. http://www.embedall.com.

[5] 李善平,刘文峰,王焕龙,等. Linux 与嵌入式系统[M]. 北京:清华大学出版社,2003.

[6] 康伟民. 基于嵌入式系统的 MiniGUI 的移植[J]. 大众科技,2007(12):100-102.

[7] 王启文,韩秀玲,孙 波. 基于 MiniGUI 的多进程图形用户界面的研究[J]. 微计算机信息,2007(8):78-80.

[8] 郑端建,郭 磊,魏世民. MiniGUI 图形库在嵌入式 Linux 平台上的移植与实现[J]. 仪表技术,2008(10):10-11.

(上接第 12 页)

Network Design Using Shuffled Frog Leaping Algorithm[J]. Journal of Water Resources Planning and Management,2003,129(3):210-225.

[7] Elbeltagi E, Hegazy T, Grierson D. Comparison among five evolutionary-based optimization algorithms[J]. Advanced Engineering Informatics,2005,19:43-53.

[8] 邵晓巍,邵长胜,赵长安. 利用信息量留存的蚁群遗传算法[J]. 控制与决策,2004,19(10):1187-1189.