第2卷第4期 管理科学学报 Vol. 2 No. 4
1999年12月 JOURNAL OF MANAGEMENT SCIENCES IN CHINA Dec. 1999

遗传规划研究与应用中的若干问题①

林 丹，寇纪淞，李敏强
（天津大学系统工程研究所，天津 300072）

摘要：简要介绍了遗传规划的基本内容，结合具体例子论述了遗传规划成功应用的实际领域的特征，指出研究与应用遗传规划时常见的两个问题，并从理论上给出了相应的对策，以期提高遗传规划的效率.

关键词：遗传规划；应用领域；冗余性；子程序
分类号：TP18 **文献标识码**：A **文章编号**：1007-9807(1999)04-0062-08

0 引 言

遗传规划(Genetic Programming，GP)是一种计算机自动编程技术，是遗传算法的延伸和扩展. 我们知道，遗传算法是模拟自然界的生物进化过程并以达尔文的进化论为指导思想的一种自适应搜索算法. 当应用遗传算法时，通常需要建立问题的搜索空间与一个二进制串(染色体)之间的一一对应，这需要对问题本身有相当深刻的了解和很好的判断，因此在一定程度上妨碍了遗传算法的应用. 在遗传规划中所使用的个体(染色体)是长度和大小可变的程序树，这种动态树状表示方式自然、灵活，更接近人类解决问题的自然方式，在很大程度上克服了遗传算法的局限性，是一种不依赖于具体问题领域特定知识的机器自动学习的软方法. 本文首先简要介绍了遗传规划的内容，然后结合具体的应用例子(其中包括我们的工作)论述了遗传规划能成功应用的实际领域的特征，指出研究与应用遗传规划时常见的两个问题，即个体大小"爆炸"现象及子程序重用性. 并从理论上给出了相应的对策，以期提高遗传规划的效率.

1 遗传规划简介

遗传规划的基本步骤如下：

(1) 确定原始函数和端点集合 F 和 T，由此随机产生初始群体，每个个体由原始函数和端点组成.

(2)在迭代停止准则满足前，反复执行下列步骤：

(A)执行每个作为个体的程序，按适应值度量给它一个适应值.

(B)将下列算子应用在以适应值为基础选择出来的程序个体上以产生新的个体.

(i)复制：将现有个体复制到新的群体中.

(ii)交叉：交换两个个体的相应的随机选定的部分.

(iii)变异：改变个体的随机选定的部分.

和遗传算法一样，交叉算子是主要算子，交叉概率 Pc 一般≥0.8，而变异算子是次要算子，变异概率 Pm 一般≤0.1.

① 收稿日期：1998-08-05；修订日期：1999-02-02.
基金项目：国家自然科学基金资助项目(69574022，69974026).
作者简介：林丹(1968-)，男(汉族)、福建省福州人，天津大学博士生、讲师.

2　遗传规划适用的应用领域[1]

2.1　适用领域的特征

遗传规划的创立者 Koza 教授对遗传规划成功的应用领域做了一个回顾总结,并对遗传规划可能的有前途的应用领域做了一个展望.

一个实际应用领域,如果具备以下若干或是全部特征,则它们就特别适合于遗传规划的应用:在一个领域中,传统的数学分析方法没有或不能提供解析结果;对相关变量之间的内在联系了解得很少或现有的理解很可能是错误的;找出问题的解的形状和大小是问题的一个主要部分;近似解是可以接受的,或者只可能得到近似解;有大量的以计算机可用的形式存在的数据需要检查、分类和集成;在性能方面的微小改进能被常规测量(或很容易衡量)并且得到高收益,那么,这个领域就很可能成为遗传规划成功应用的领域.例如,自动控制方面的问题就特别适合于遗传规划,因为传统的数学方法无法对许多有实用价值的问题提供解析解,控制工程师们也愿意接受近似解,并且在性能方面的微小改进都具有重大的价值.在某些领域中,数据以机器可用的方式大量堆积,象生物学中的 DNA 序列的数据、天文观测,地质与石油的数据,金融时间序列数据,卫星观测数据、天气数据,营销数据库等.这些领域中的问题就成了遗传规划特别有意义的潜在的实际应用领域.

当前,遗传规划已经能够解决非平凡的实际问题,通过进化的过程机器进行自动学习得到的解答与人类在相同问题上通过人的大脑的智力活动得到的结果相当,甚至在一些具体问题上更理想.在细胞自动机,人造卫星控制,分子生物学以及电子线路的设计等广泛的领域中,用遗传规划进化得到的计算机程序的结果就是如此.在这里,介绍两个有关的例子.

细胞自动机的程序设计是一件相当困难的任务,特别是当所希望的计算需要经过细胞空间中的较长距离进行局部信息的通讯和集成时尤其如此.对于用一维两个状态的细胞自动机的主群分类任务,在二十多年中给出了许多的规则,包括一些用机器自动学习得到的结果.用带自动定义函数的遗传规划得到的规则,其精度为 82.326%,超过了以往的任何规则.与它们不同的是,这个遗传进化得到的规则更多地利用了关于区域和质点的内含特性来达到在细胞空间中表示信息和进行信息通讯的目的.

遗传规划的另一个成功的应用例子是设计满足用户特定要求的电路,包括它的拓扑结构和所有器件的值的大小.在自动设计纯粹的数字电路方面已取得了不少的进展,但是在自动设计模拟电路和混合的数字—模拟电路方面则进展甚微.用遗传规划可以设计出一种公认难以设计的滤波器.这种滤波器的拓扑结构是相当复杂的,人们以往的设计方案的效果十分不理想,而用遗传规划得到的设计方案很好地满足了设计要求.详细的介绍可见[2].

2.2　我们的一些工作

我们遗传规划课题组承担了国家自然科学基金项目"遗传规划的理论与应用",在遗传规划研究方面做了一定的工作并取得了一些结果.现将两个有代表性的应用例子介绍如下.

在土木工程中,碎石桩复合地基是目前处理软弱地基的常用方法之一.由于此方法在我国应用的历史比较短,以及作为地基土的物理和力学性能比较复杂,人们对碎石桩复合地基理论的研究尚处于初期阶段.目前,在工程中所采用的是根据研究人员及工程技术人员自己的经验及有关理论提出的碎石桩复合地基承载力的计算公式,这些公式的精度是不相同的.

采用遗传规划方法,利用文献中的数据,进行地基承载力计算的拟合.根据地基处理的具体情况,经过与有关专家探讨,以及参考现有经验公式等资料,应用遗传规划得到了拟合公式.用拟合公式进行计算的结果与现在较常用的三个经验公式进行比较,结果表明,这个公式的精度比其中的两个方法高,所以这是一种可行的,值得研究的有效算法.同时,对遗传规划在公式拟合中的收敛性进行了分析,严格证明了遗传规划拟合得到的公式在一定条件下以概率 1 逼近真实模型,即当遗传规划的代数趋向无穷时,遗传规划拟合公式产生的误差大于指定精度的概率趋于零.因此,遗传规划是一种值得推广使用的有效算法,它避免了用传统方法建模时,事先选定未知模型类型的盲目

性[3,4].

这个例子之所以成功,关键在于遗传规划所应用的领域正好符合前文所说的遗传规划能成功应用的特征.比如,目前没有合适的数学工具推导出一个统一的公式,推导经验公式需要处理大量的实际工程数据等.

另一个应用的例子是关于图论中的欧拉图问题.所谓的欧拉图是指一个无孤立点的图,其中存在一条经过图中每条边一次而且只一次的欧拉回路.很多实际问题都可以化为有向或无向欧拉图问题.例如,编码器的设计就可以转化为求有向图的欧拉回路问题.

我们所做的工作是对任意给定的图,如果有欧拉回路存在,则用遗传规划方法求出该图的一条欧拉回路.分别用带自定义函数和不带自定义函数的遗传规划方法进行计算,结果表明,不管是哪一种方法,都能很快地求出解来,而带自定义函数的遗传规划方法在性能方面要更好一些[5].

特别需要指明的是,我们假定不知道任何有关欧拉图的知识,在编程中没有用到任何有关欧拉图的性质,而是让遗传规划在进化过程中自己去发现和利用欧拉图的内在性质.事实说明,采用带自动定义函数的遗传规划方法,是一个高效的不依赖于具体应用领域的专门知识的普适的机器自动学习方法,具有广泛的应用领域和大好前景.

3 遗传规划中个体大小"爆炸"现象及其对策

在遗传规划中存在着熟知的所谓"规模爆炸"问题,亦即作为程序个体表示的文法树的大小,包括它的深度和广度,在进化过程中不断地增大,而与此同时作为解的群体却没有任何改进.这一现象为许多研究者所发现.对于大多数的应用问题来讲,树不能太大,否则很难将它翻译为一个有意义的结果;同时,正如许多文献所指出的那样,"爆炸"现象常常伴随着优化进程变慢的表现.

3.1 内含子的作用与危害[6~9]

在程序树规模变大问题中,可以观察到一个十分有趣的现象,这就是"内含子"(introns)大量产生的问题,也可称之为"冗余"(redundancy)问题.图 1,2 是两个冗余的程序树的简单例子.在图 1 中,因为第一个变量的值为 false,所以"与"运算的结果恒为 false,而不管第二个变量的值是什么.因此,如果有一个子树以节点 3 为顶点的话,那么它对程序树所代表的整个程序的最终结果没有任何影响.图 2 中,实际计算结果是 $(x\times 3)\div 3=x$.类似的现象在符号回归问题的求解过程中是非常普遍的.为了压缩这种冗余的语句,Koza 不得不对最后的结果使用自动编辑器来消除这种现象.但是,这仅仅是在最后一步所作的工作,对于在进化过程中怎样减少该现象并没有提供帮助.

内含子是一个遗传学术语,指生物染色体上的某些不表达为蛋白质的基因片段,它们只是提供了有意义的基因(外显子)之间的间隙.文献[6]首先将内含子的概念引入遗传算法中,在一个串中的某些基因座上放置与适应值计算无关的等位基因,或干脆认为这些位置是空的.引入内含子的意义在于,进行交叉操作时,当交叉点的位置落在内含子上时,不会破坏已有的有意义的串,因此能有效地保护现有的模式.

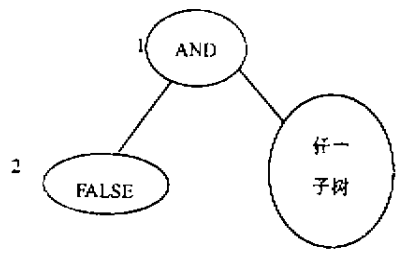

图 1 冗余程序树例 1

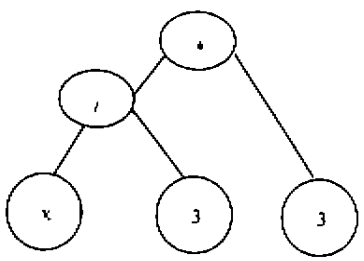

图 2 冗余程序树例 2

Nordin[7,8]模仿遗传算法的串表示,引入了线形表示的程序个体,而不是用通常的树状表示,并且在遗传规划中引进了内含子的概念,亦即"对任何适应值函数,都不影响所在程序的表现的程序块".在实例中发现,进行了交叉操作以后,有相当的个体的适应值下降了,同时有相当的个体的适应值保持不变.经过分析得知,很多个体通过增加它所包含的内含子的个数来保护自己不受交叉操作破坏作用的影响而保持适应值基本不变.

Nordin 的分析是基于他的程序个体的线形表示的,不完全适用于通用的遗传规划.对于如何在程序树中引入内含子的概念以及做出相应的分析,仍是一个有争议的问题.一个比较自然的定义是:称个体树中的某个节点是内含子,如果该个体的适应值与顶点在这个节点的子树无关.比如在上面的图 1 中,节点 3 就是内含子.显然,如果一个节点是内含子,则以它为顶点的子树中的任何节点都是内含子.由于动态树状表示的复杂性,对之进行理论分析是十分困难的(这也是遗传规划理论研究中的主要困难所在),现有的结果也是很初步的.

如前所述,内含子的出现与存在,在保护个体不受交叉操作的破坏方面是有益的,但是一个程序中有过多的内含子会造成程序树过度膨胀,影响运算效率,同时也使得对程序意义的解释变得困难.已经有多种方法来处理这种程序树的大小"爆炸"的问题,其中最简单的方法是在遗传规划的运行过程中,限制树的大小,比如将树的最大深度限制为 10.这种方法的问题是群体的多样性得不到保证.为了保证在群体中有足够的遗传物质,只好使用很大的群体规模,必然大大增加计算量.更重要的是,在多数应用问题中,事先根本不可能确定应当如何选择树的大小才能解决问题.取得太大,没有意义,而取得太小时,有可能得不到正确的解.

3.2　消除冗余性的方法

普遍的方法是引入类似于优化问题中"罚函数"项的简约压力项(parsimony pressure).对于一个最小化问题,这就相当于将一个与树的大小有关的项加到原始的适应值函数上以产生新的适应值函数,即

$$f(I) = O(I) + \delta C(I) \tag{1}$$

这里 I 表示一个个体,$O(I)$ 为 I 的原始适应值,$C(I)$ 给出了 I 的树状表示的复杂度,如树的深度,节点的个数等,δ 为简约压力因子,$f(I)$ 为新的适应值函数.这种方法的困难在于如何权衡简约压力因子 δ 的大小.如果 δ 太小,则对于减少内含子数目的效果并不理想;而 δ 太大,遗传规划倾向于选择短的解,即使短解的原始适应值 $O(I)$ 较差,这就会导致收敛到错误的解.

由于事先无法估计最优解的规模,δ 的确定当然是以自适应选择为佳,亦即在进化过程中自动地调整 δ 的大小.在文[10]中一种称为"自适应平衡精确度和简约性"(adaptive balancing of accuracy and parsimony)的方法是根据当前最优解来自适应地调整 δ 的值.有趣的是,这种方法不需要遗传规划交叉操作的分析,其中的启发式策略是由统计分析所导出的.

有的作者认为上述方法虽然有时是有好处的,但是由于它经常要干涉群体的适应值,因此在大多数的例子中,该方法的效果并不理想.可以采用一种称为"踪迹开采"(trace mining)的技术,即引入一种机制来跟踪子树出现的频率和显著性(即对其所在个体树的适应值的影响).这些信息被用来选"较好"的子树来进行交叉,以获得比较高的收敛性[11].这一方法的优点是它既不影响遗传规划的结构,也不影响其过程.但是,显然这种机制的实施需要花费巨大的额外的计算代价.

文献[12]中提出用"个体目标转换"(individual objective switching)的技术来解决这一问题.这是一种适合在进化计算领域中处理约束问题的普遍方法.现在将求得一个既短又精确的解的问题表示为下述优化问题:

$$\begin{aligned} &\min C(I) \\ &\text{s.t.}\ O(I) < \varepsilon \end{aligned} \tag{2}$$

优化的目标是:在个体的原始适应值小于某个事先指定的限制时,最小化树的规模.因此,适应值函数可以定义为

$$f(I) = \begin{cases} 1 + O(I) & : O(I) \geqslant \varepsilon \\ C(I)/C_{\max} & : \text{else} \end{cases} \tag{3}$$

这里 $C_{\max}$ 为事先指定的树的最大规模.

这个方法突出的优点是它不需要去确定、选择简约压力因子 δ 来平衡解的精确度和大小.但是,如何在不了解最优解的性状的情况下确定 ε

是一个棘手的问题.

在实践中发现,上述各种方法各有其适用的具体问题.但如何找出问题的特征以便事先就能判断使用哪一种方法更合适,仍是一个没有解决的问题.

4 子程序的重用性和共享性

在大多数的计算机语言中,都有由若干语句组成的相对独立的小程序块,完成一定的功能,可以重复地被程序的其他部分和其他程序所调用.这就是大家熟悉的过程(Pascal 语言)、子程序(Fortran 语言)或函数(C、Lisp、Fortran、Pascal 语言)等.不妨将它们通称为"子程序".对于一名编程人员来说,经常采用的是自顶向下的分析方法,将一个大的任务分解为多个相对独立的子任务,然后针对每个子任务编写相应的子程序,最后调用这些子程序单元来完成整个功能.由于子任务还可以分解为相对子任务,即子任务调用也是层次的,一个子程序能调用别的子程序,同时又能被其他子程序调用.能重复使用和为其他程序共享的子程序的存在和特性,在程序设计中起至关重要作用,这些都应当反映在作为自动程序设计的遗传规划中.

4.1 自动定义函数方法[13~21]

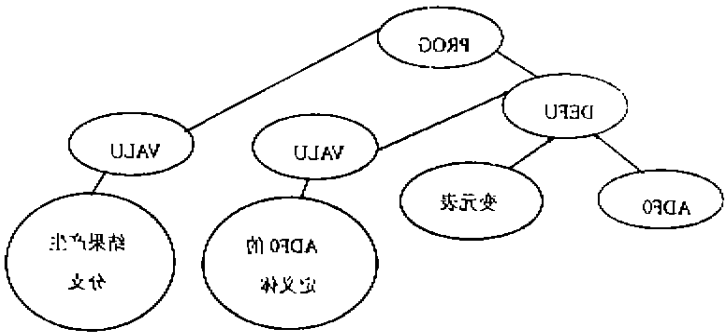

图 3 一个自动定义函数的例子

Koza 在[20]中首次全面介绍了遗传规划方法,子程序的重用性已经占有一定篇幅,而在两年以后出版的专著[13]中,子程序的重用性则成为主题.在这本书中介绍的自动定义函数(automatically defined functions, ADFs)方法很好地解决了子程序的重用问题.图 3 是带有一个函数定义分支和一个结果产生分支的计算机程序的简单例子,它是使用自动定义函数的遗传规划群体中的一个个体.在左边的函数定义分支中,定义了一个自动定义函数 ADF0,右边的结果产生分支可以调用它.

当一个个体的函数定义分支中所定义的自动定义函数在进行层次调用时,只能调用本个体函数定义分支中所定义的自动定义函数.类似地,结果产生分支也只能调用本个体的函数定义分支中定义的自动定义函数.换句话说,一个个体不能调用其他个体的任何自动定义函数,同时该个体的任何自动定义函数也不能被其他个体所调用.这在一定程度上限制了子程序的共享性.而当群体进化到接近最优解时,某些能够很好地解决子问题的自动定义函数已经通过进化而产生,若能为整个群体中的个体所调用(不论是结果产生分支还是自动定义函数),将必然加速收敛过程.同时,由于遗传规划所特有的树状表示方法,使得两个自动定义函数在交叉之后基本上不可能保持不变.因此,已经进化产生的优秀的自动定义函数不可避免地遭到破坏,从而可能导致收敛的振荡现象发生.由此可见,适当地将优秀的自动定义函数通过某种方式保护起来,保证其不被遗传算子破坏,是一个值得考虑的问题.

4.2 模块方法

事实上,模块(module)方法[14,15]就是基于这种考虑而提出的.在模块方法中,定义了一个基因库(genetic library)来放置模块,这些模块是全局

定义的,作为一个任务的特定分解的某些子任务的解.这样,这些模块能被群体中的所有个体共享,而且在进化过程中不会被交叉算子破坏,起到保护子任务的优秀解的作用.

模块的形成过程是从个体中随机地选取一个子树,通过压缩算子的作用,将它压缩成一个模块,并将它加入基因库中.压缩算子有深度压缩(depth compression)和叶压缩(leaf compression)两种算子,图 4 给出深度压缩算子作用过程的示意图.

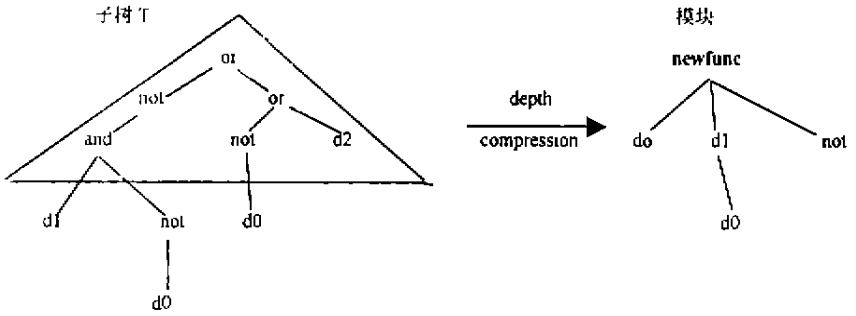

图 4　一个深度压缩的例子

从上图可以看出,一个子树 T 在深度压缩算子作用下成为一个新定义的函数,亦即模块 newfunc,并将之放入基因库中,供整个群体调用.生成模块 newfunc 后,原来的子树就用模块的符号名称表示,在树中仅仅占有一个节点,因此 T 的内容不会被交叉操作所破坏,同时也不允许变异操作直接作用于其上,从而有效地保护了压缩前的子树 T 的内容.

另外一个称为模块扩张(expand)的算子可以将某个引用了模块的个体中的所有模块都扩张为原来的子树形式,从而产生一个不含任何模块引用的新个体.模块扩张使得一个模块中的内容重新参与交叉等遗传操作.

一个模块被定义并放入基因库以后,它就不再进化.必须注意的是获取模块的速度应当使得基因库中的模块个数保持在一定的范围内.

模块方法和自动定义函数方法的一个区别是基因库中的模块作为一个群体是固定不变的,其个体不存在进化.而自动定义函数作为一个群体,始终随着整个结果产生分支群体在不停地进化.另一个区别是,自动定义函数的个数、叶结点和端点类型都是事先确定的,而模块是在进化过程中随机地获取的.从这一点上看,自动定义函数方法在自适应方面有所欠缺.

值得一提的是,Koza 在[16]中提出了基于自动定义函数的变结构操作(architecture-altering operations),该操作能够在进化的过程中自动地引入或删除函数定义分支,调整每个函数定义分支的变量的个数,改变函数定义分支之间的层次调用关系.这一做法大大提高了自动定义函数方法的自适应性.Koza 就是将带变结构操作的自动定义函数方法应用于集成电路的自动设计这一棘手的问题,取得了非常理想的结果,充分显示了该方法的强大威力.

实际计算表明[17],模块方法在所考虑的具体问题中,最后的计算结果的各项性能都不及自动定义函数方法.这也许是因为模块方法中基因库里的模块群体没有进化的缘故.但是不管怎样,其思想方法还是很有启发性的.

4.3　共享自动定义函数方法

日本学者提出的共享自动定义函数方法(sharing－ADFs[23,24])从某种意义上说是将上述两种方法有机地结合在一起.它是在自动定义函数方法的基础上,通过引入与模块方法中的基因库相类似的概念—库而得到的.在全体自动定义函数中根据一定的规则抽取出若干自动定义函数放置在库中,库中所有的自动定义函数都可以由整个进化群体中的所有个体(包括自动定义函数和结果产生分支部分)所调用.这就是"共享"一词的含义.与此同时,库中的所有自动定义函数组成一个小的群体,也随着进化群体而进化.库中每个

个体 SR_i 的适应值定义如下：

$$fitness_{SR_i}=\frac{\sum fitness_k}{ref_{SR_i}}$$

这里，$fitness_k$ 是进化群体中调用了库中的个体 SR_i 的个体 k 的适应值，ref_{SR_i} 是所有调用了 SR_i 的个体的总数. 适应值最差的 SR_i 就可能由一个新得到的自动定义函数所代替.

共享自动定义函数方法综合了自动定义函数方法与模块方法的优点，同时又克服了两种方法的不足，应该说是一种很有前途的好方法. 从文献[23,24]中给出的算例来看，用自动定义函数方法进行计算，当群体趋向于收敛时，出现了前面提到的振荡现象. 而用共享自动定义函数方法求解，就没有这种现象，同时其他性能也有不同程度的提高. 但是，目前该方法应用的实际例子还不是很多，其有效性仍然有待进一步的验证.

需要说明的是，在具体计算时，通常将库中的自动定义函数的个数限制在10以内，因此与一般的自动定义函数方法相比，并没有增加很多的计算量.

5 结论

在前文中讨论了遗传规划具有良好应用前景的实际领域的特征，并且指出研究与具体应用遗传规划时常见的两个问题，从理论上给出了相应的对策. 我们相信，选择适宜的应用问题，并注意在求解过程中采取合适的对策以避免冗余现象的发生以及提高程序的可重用性和共享性，一定能取得满意的效果.

参 考 文 献：

[1] 林 丹，李敏强，寇纪淞. 遗传规划的应用领域问题[C]. 中国系统工程1998年会论文集，343～347

[2] Koza J R. Future work and practical applications of genetic programming[M]. In: Handbook of Evolutionary Computation，Oxford University Press，1997. 123～134

[3] 寇纪淞，马丰宁，李敏强. GP在搜索技术中的应用[J]. 管理科学学报，1999，2(2)：35～40

[4] 李书全，寇纪淞，李敏强. 遗传规划及其在地基设计中的应用[J]. 系统工程学报，1997，4(12)：85～92

[5] 马丰宁，寇纪淞，李敏强. 用遗传规划求欧拉回路[J]. 系统工程理论与实践，1997，17(5)：19～29

[6] Levenick J. Inserting introns improves genetic algorithm success rate：Taking a cue from biology[M]. In: Rawlins，ed. Proceedings of ICGA'91. CA：Morgan Kaufmann Publishers Inc.，1991. 123～128

[7] Nordin P，Banzhaf W. Complexity compression and evolution[M]. In: Eshlman D，ed. Proceedings of ICGA'95，CA：Morgan Kaufmann Publishers Inc.，1995. 310～317

[8] Nordin P，Francone F，Banzhaf W. Explicitly defined introns and destructive crossover in genetic programming[R]. In: P. Rosca，ed. Proceedings of the workshop on GP. Technical Report 95.2，University of Rochester，1995. 6～22

[9] Blickle T. Evolving compact solutions in genetic programming: A case study[C]. In：Schwefel H-P，ed. Proceedings of the International Conf. PPSN IV，1996. 564～573

[10] Zhang B-T，Muhlenbein. Balancing accuracy and parsimony in genetic programming[J]. Evolutionary Computation，1995，3 (1)，17～38

[11] Tackett W. Mining the genetic program[J]. IEEE Expert，June 1995，10(3)，451～466

[12] Blickle T，Thiele L. A mathematical analysis of tournament selection[C]. In：Eshlman D，ed. Proceedings of the ICGA'95，Morgan Kaufmann Publishers Inc.，1995. 9～16

[13] Koza J R. Genetic Programming II[M]. Cambridge，MA：The MIT Press，1994.

[14] Angeline P，Pollack J. Evolutionary module acquisition[C]. In: Fogel D，ed. Proceedings of the Second Annual Conference on Evolutionary Programming. CA：Evolutionary Programming Society，1993. 263～274

[15] Angeline P，Pollack J. The evolutionary induction of subroutines[R]. In: Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society. Lawrence Erlbaum Associates Inc.，1992. 371～377

[16] Koza J R. Use of automatically defined functions and architecture-altering operations in automated circuit synthesis using genetic programming[C]. In：Proceedings of the First Annual Conference on Genetic Programming. The MIT

Press,1996. 120～129

[17] Kinnear K E. Alternatives in automatic function definition[M]. In: Kinnear KE,Ed.. Advances in Genetic Programming. The MIT Press, 1994. 119～143

[18] Andre D, Teller A. A study in program response an the magive effects of introns in genetic progamming[C]. In: Proceedings of the First Annual Conference on Genetic Programming, The MIT Press,1996. 231～239

[19] Angeline P J. The effects of noise on self-adaptive evolutionary optimization [C]. In: Fogel L J, Angeline P J and Baeck T. Eds. Proc of the Fifth Annual Conference on Evolutionary Programming. Cambridge, MA: MIT Press. 1996. 441～450

[20] Angeline P J. Adaptive and self-adaptive evolutionary computations[C]. In: Palaniswami M, Attikiouzwl R and Marks R et al ,eds. Computational Intelligence: A Dynamic System Perspective. Piscataway,NJ: IEEE Press. 1995. 152～163

[21] Spears W M . Adapting crossover in evolutionary algoriathms[C]. In: McDonnell J R, Reynolds R Fogel D B. Eds. Proceedings of the Fourth International Conference on Evolutionary Programming. Cambridge, MA: The MIT Press,1995. 367～384

[22] Koza J R. Genetic Programming I[M]. Cambridge, MA: The MIT Press , 1992

[23] Hondo N. Sharing and refinement for reusable subroutines of genetic programming[C]. In:Fogel D,ed. Proceedings of ICEC'96, NJ:IEEE Press, 1996. 431～438

[24] Hondo N. Robust GP in robor learning[C]. In:Schwefel H-P,ed. Proceedings of the International Conference PPSN IV. Berlin:Spring-Verlag,1996. 365～372

On some problems of researching and applying genetic programming

LIN Dan, KOU Ji-song, LI Min-qiang

Institute of Systems Engineering, Tianjin University, Tianjin 300072

Abstract: Genetic Programming is increasing the popularity as the basis for a wide range of learning algorithms. In this paper a brief introduction to Genetic Programming (GP) and the characteristics of the areas in which GP can be successfully applied are presented, combined with some examples including some of our works. Two common problems encountered when applying GP, i. e. the exploration of the size of individuals and the available subroutines, are discussed and the methods to solve them are given in order to improve the power of GP.

Keywords: genetic programming; application area; redundancy; subroutine