文章编号:1001 - 9081(2007)08 - 1944 - 04

基于带状态回溯个体进化的子结构发现

常新功[1,2],寇纪淞[1],李敏强[1]
(1. 天津大学 管理学院,天津 300072; 2. 山西财经大学 信息管理学院,太原 030006)
(c_x_g@126.com)

摘 要:将进化算法引入图数据挖掘,以克服贪婪式查找易陷入局部极值的问题。针对图数据挖掘中经常遇到的子图同构问题,提出了带状态回溯个体的概念,从而使遗传算子的设计更为合理。另外,还提出了一种新的多样性保持方案,从种群的组成和个体的生成两个方面提高了种群的多样性。在进化过程中随时去掉当前种群中没有潜力的个体的机制使查找空间缩小了一半。实验结果表明,以上措施增强了算法的寻优能力,提高了算法的效率和解的质量。

关键词:进化算法;图数据挖掘;子结构发现;最小描述长度
中图分类号: TP18 **文献标志码**:A

Evolutionary substructure discovery algorithm based on individuals with state backtracking

CHANG Xin-gong[1,2], KOU Ji-song[1], LI Min-qiang[1]
(1. *School of Management, Tianjin University, Tianjin* 300072, *China*;
2. *Faculty of Information and Management, Shanxi University of Finance and Economics, Taiyuan Shanxi* 030006, *China*)

Abstract: In order to overcome the limit that the greedy search, which is often used by some prevalent graphical data mining systems, may often end up by providing sub-optimal solutions, an evolutionary algorithm was imported to perform data mining on databases represented as graphs. In order to handle the subgraph isomorphism problem, the concept of individuals with state backtracking was proposed, where the state information of some potential individuals produced during the evolution were preserved, which contributed to the design of genetic operators. A new diversity keeping scheme was also proposed to preserve the diversity both from the composition of population and the way of generation of individuals. In addition, removing the individuals not potential in current population reduced the search space greatly. Experimental results show that these measures enhance the searching capability of the algorithm and hence increase both the efficiency of the algorithm and the quality of solutions.

Key words: Evolutionary Algorithms (EA); graphical data mining; substructure discovery; Minimum Description Length (MDL)

近年来,从图数据中发现典型子(图)结构的问题得到了众多的关注和研究,与其相关的图数据挖掘算法已广泛应用于化学、生物学、计算机网络、WWW 和社会网络分析(Social Network Analysis, SNA)等众多领域。这些算法可以分为两类:一类基于频繁模式发现,可以找出满足最小支持度约束的所有子结构,其中代表性的有基于图论和 Apriori 算法的 AGM[1]、FSG[2,3] 和 gSpan[4],基于归纳逻辑程序设计(Inductive Logic Programming, ILP)的 WARMR[5],基于归纳数据库的 MolFea[6]等;另一类可以找出不仅是频繁的而且更有意义、更典型的子结构,其中基于柱状查找(Beam Search)和最小描述长度(Minimum Description Length, MDL)启发式策略的 SUBDUE[7]是这类算法中的典型代表。本文提出的算法也属于第二类。鉴于 SUBDUE 采用贪婪式的查找方式,易于陷入局部极值问题,本文引入了擅长全局寻优的进化算法,以期获得全局近似最优解。图数据挖掘不可避免地会遇到子图同构问题,为此本文提出了带状态回溯个体的概念,以保存一个个体进化过程中有发展潜力的祖先个体的信息,从而使遗传算子的设计更为合理。另外还提出了一种基于个体年龄和生成方式的多样性保持方案。实验表明,以上措施对解的质量和求解效率有明显提高。

1 基本概念

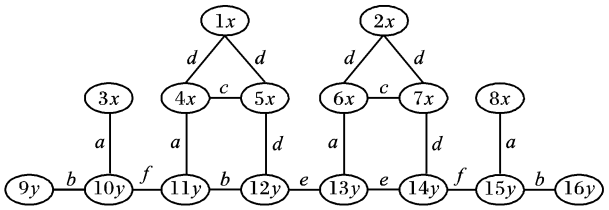


图 1 一个样本图

许多问题均可转化为带标签的图,它是一个四元组,$G = (V,E,L,l)$,其中,V 为节点集,$E \subseteq V \times V$ 为边集,L 是标签集,标签函数 $l: V \cup E \rightarrow L$ 赋与节点和边不同的标签(label)以区分其所代表的对象类型或对象间关系类型的不同。图 1 是一个带标签的图,其中,椭圆代表节点,数字为节点的编号,字

收稿日期:2007 - 02 - 13;**修回日期**:2007 - 04 - 26。 **基金项目**:国家自然科学基金资助项目(70571057)。
作者简介:常新功(1968 -),男,山西太原人,副教授,博士研究生,主要研究方向:进化计算、数据挖掘; 李敏强(1965 -),男,河北人,教授,博士生导师,主要研究方向:进化计算、机器学习; 寇纪淞(1947 -),男,贵州人,教授,博士生导师,主要研究方向:进化计算。

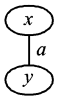
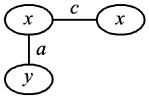
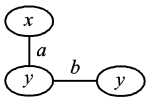
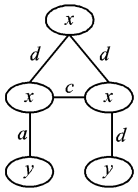
符 x、y 为节点标签。线段代表边,a、b、c、d、e、f 为边标签。

1.1　子结构及其实例

重复出现的数据片段容易引起人们的关注,而这些数据片段的公共结构正是我们要寻找的子结构。图数据中所有彼此同构的连通子图的结构(和节点编号无关只和 label 有关的节点互连关系)称为一个子结构,而这些连通子图称为该子结构的实例。

表 1 给出了图 1 中的几个子结构及相应的实例。其中 $g(n1, n2,\cdots)$ 表示图中由节点 $n1,n2,\cdots$ 构成的一个子图。例如 $g(3,9,10)$, $g(4,11,12)$, $g(8,15,16)$ 为彼此同构的连通子图,其公共结构为 $S3$。在不引起混淆的情况下,为了表示方便,在下文中还将以 $S(\alpha\beta\cdots)$ 的方式表示一个子结构,其中 $\alpha,\beta,\cdots$ 为边标签,例如,$S1$、$S2$、$S3$、$S4$ 也可分别表示为 $S(a)$、$S(ac)$、$S(ab)$、$S(acddd)$。子结构和实例的关系就像是面向对象方法中类和对象的关系,子结构是其实例的公共模板,每个实例是其子结构在图中的一次出现。

表 1　图 1 中的几个子结构及其实例

编号	子结构	实例
$S1$		$g(3,10)$ $g(4,11)$ $g(6,13)$ $g(8,15)$
$S2$		$g(4,5,11)$ $g(6,7,13)$
$S3$		$g(3,9,10)$ $g(4,11,12)$ $g(8,15,16)$
$S4$		$g(1,4,5,11,12)$ $g(2,6,7,13,14)$

1.2　MDL 与子结构的评价

本文和 SUBDUE 均以最小描述长度 MDL[8] 作为一个子结构的评判准则。MDL 基于如下思想:数据中规则性的东西可以用来压缩该数据,规则性越强,压缩程度越大。子结构就是这样一种规则性的东西,可用其实例压缩数据。设 G 代表一个图,S 是它的一个子结构,在用了子结构 S 之后 G 的描述长度为 $DL(S)+DL(G|S)$,其中 $DL(S)$ 是子结构 S 的描述长度,$DL(G|S)$ 表示在将 G 中 S 的每个实例分别换为一个代表该子结构的新节点后,所得图的描述长度。本文采用和文献[9]中同样的方式,取 $DL(S)=\mathrm{Size}(S)$,即 S 的节点个数与边数之和,$DL(G|S)=\mathrm{Size}(G|S)$,即将 G 中 S 的实例换为节点后所得图的节点个数和边数之和。子结构 S 对 G 的压缩率定义为$(\mathrm{Size}(S)+\mathrm{Size}(G|S))/\mathrm{Size}(G)$。以图1为例,$\mathrm{Size}(G)=16+19=35$,$\mathrm{Size}(S3)=3+2=5$,$\mathrm{Size}(G|S3)=10+13=23$,压缩率 $=(\mathrm{Size}(S3)+\mathrm{Size}(G|S3))/\mathrm{Size}(G)=(5+23)/35=0.8$。根据 MDL 原理,用了子结构 S 之后 G 的描述长度越小,即压缩率越小,S 对 G 的压缩程度越大,子结构 S 的规则性就越强。寻找压缩率最小的子结构正是本文的目的所在。实践和理论证明,基于 MDL 的方法能够找到更为关键、更有意义的子结构[7]。

1.3　子结构的扩展

包括本文在内的所有图数据挖掘算法,其子结构都是按节点数或边数从小到大逐步生成的。一个子结构加一边或一边又一点得到另一个子结构的过程称为子结构的扩展。前者是后者的父亲,后者是前者的孩子。对一个子结构 Sub 扩展的具体过程是:1)对 Sub 的所有实例以所有可能的加一边的方式进行扩展;2)所得结果插入到一个实例列表中;3)对该列表中的实例按同构关系进行归类,不同的类对应不同的新的子结构;4)返回由这些子结构构成的列表。例如对 $S1$ 进行扩展,$S1$ 有四个实例,对 $g(3,10)$ 以加一边的方式扩展可得 $g(3,10,9)$ 和 $g(3,10,11)$,对 $g(4,11)$ 扩展可得 $g(4,5,11)$、$g(4,10,11)$ 和 $g(4,11,12)$,同理扩展 $g(6,13)$ 可得 $g(6,7,13)$、$g(6,12,13)$ 和 $g(6,13,14)$,扩展 $g(8,15)$ 可得 $g(8,14,15)$ 和 $g(8,15,16)$。对这些新得的实例按同构关系归类,如 $g(4,5,11)$ 和 $g(6,7,13)$ 为一类,$g(3,10,11)$ 和 $g(8,14,15)$ 为另一类 …… 每一类对应到一个子结构,最后可得子结构列表:$S(ac)$,$S(ab)$,$S(ae)$,$S(af)$。

1.4　子图同构和子结构查找的单向性

设 G 和 H 是两个带标签的图,如存在函数 $f:V(G)\rightarrow V(H)$ 满足如下三个条件,则称 G 和 H 同构:1)f 是一个双射;2)$\forall u\in V(G), l_G(u)=l_H(f(u))$;3)$\forall(u,v)\in E(G)$, $(f(u),f(v))\in E(H)$ 且 $l_G(u,v)=l_H(f(u),f(v))$。在 H 中求所有和 G 同构的子图的问题称为子图同构问题,它是一个 NPC 问题[10],没有多项式级的算法存在。

子结构的边数从少到多逐步扩展生成,评价的过程就是子结构的查找过程,它具有单向性,即子结构只能从小到大生成,而不能从大到小生成,因为不能保证小子结构的实例都可以从大子结构的实例中得到,这是由子图同构问题造成的。例如 $S1$ 可以通过扩展生成 $S3$,但反之是行不通的,如要通过减边的方式由 $S3$ 生成 $S1$,$S3$ 有三个实例,通过减边,只能生成 $S1$ 的三个实例,$g(3,10)$、$g(4,11)$ 和 $g(8,15)$,第四个实例 $g(6,13)$ 只能通过在整个图中查找得到,这就不可避免地要做子图同构。子图同构问题是造成问题复杂性的根本原因所在。

2　进化子结构发现算法

进化算法(Evolutionary Algorithms, EA)是一种基于自然选择和遗传变异等生物进化机制的全局性随机搜索算法[11]。它从代表候选解集的初始种群开始,迭代地进行选择、交叉、变异等遗传操作,最终发现最优解或近似最优解。除具有普适性、可伸缩性、并行性和鲁棒性等特点外,它还具有群体搜索的特点,即它是从一群而不是单一的初始点开始沿多条路径搜索,这使其可以有效的跳出局部极值点。

2.1　带状态回溯的个体

为了克服子结构查找的单向性,引入了个体的历史的概念以保存一个个体(即子结构)从小到大进化过程中较优的祖先个体的状态信息,以便能够回溯查找,从而使查找过程具有部分的可逆性。将一个个体的历史组织成邻接表的形式,以便在存入祖先信息的同时还能维护历史中各子结构的祖先关系。历史中的子结构不再带历史,它的部分祖先子结构可通过以上的祖先关系找到。用户可以指定历史的宽度(即邻

接表的长度),按子结构的压缩率以优胜劣汰的方式维护历史中子结构的插入和删除。图 2 的左边是一个个体,易见 $S1$、$S2$、$S3$、$S4$ 都是它的祖先子结构,设历史宽度为 5,该个体的历史可表示为右边的邻接表。这种存储方式既便于从历史中的子结构数组中按适应值进行选择,也便于进一步维护经遗传算子生成的新个体的历史。

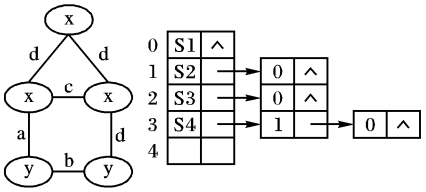


图 2 一个个体及其历史邻接表

2.2 个体的表示

染色体代表候选解,因此一个染色体就是一个子结构。一个子结构中应该包括:1)一个子结构也是一个反映节点互连关系的图(见表 1 第 2 列),因此该图的节点和边信息应包含在该子结构中;2)SUBDUE 还将一个子结构的实例信息加入到该子结构中,构成了带实例的子结构。这是一种行之有效的对付子图同构的方法。子图同构就是在一个图中查找所有和另一个图同构的子图,它是 NP 完全问题。为避免这种查找,可将实例"随身带着"。将实例信息加入到子结构中后,在对这个子结构进行扩展以生成新的子结构时,只需对其每个"随身带着"的实例进行扩展即可,而不用在整个图中去寻找新子结构的所有实例,从而大大缩小了查找空间,极大地提高了效率。

此外,一个子结构中还应包括目前找到的实例个数、目前存放的祖先子结构个数、该子结构的压缩率等信息。综上所述,一个染色体中包含了许多信息,有简单信息,也有反映结构关系的非线性信息,有定长信息,也有非定长信息(如实例列表)。因此本文没有用线性的 0 - 1 串来表示一个染色体,而是用 C 语言中的结构体来表示一个子结构,里面含有上述各项信息,用指向结构体的指针表示一个染色体。这种表示直观自然,且便于执行各种遗传操作。

2.3 种群的初始化和适应值

首先生成所有像表 1 中 $S1$ 那样的单边结构,然后从中可重复地随机选择 *popsize*(种群大小)个构成初始种群。一个子结构 S 的适应值定义为该子结构的压缩率 $(\mathrm{Size}(S) + \mathrm{Size}(G \mid S))/\mathrm{Size}(G)$。

2.4 变异

像遗传算法(Genetic Algorithm, GA)中的变异可以将 1 变为 0 或 0 变为 1,本文中的变异包括加边变异和减边变异。

2.4.1 加边变异

加边变异的过程是:设 Sub 为一个子结构,首先按 1.4 中子结构扩展方法扩展 Sub 得到一个子结构列表;在这些新的子结构中任选 n 个,其中适应值最大的记作 newSub 并作为结果返回。另外还要将 Sub 及 Sub 中的祖先子结构信息保存到 newSub 历史中。例如对 $S1$ 加边变异,扩展 $S1$ 可得四个子结构:$S(ac)$,$S(ab)$,$S(ae)$,$S(af)$。设 $n = 2$,任选的两个子结构为 $S(ac)$,$S(ae)$,其中适应值较大的 $S(ac)$ 即 $S2$ 作为结果返回。

上述的 n 是一个正整数,当 $n = 1$ 时变异操作是纯随机性的变异,n 越大贪婪性越强。这种机制将爬山算法的思想引入到了 EA 中,以缓解 EA 局部搜索能力不强的问题。它对算法的运行效率和运行结果都有较大地提高。

2.4.2 减边变异

将一个子结构变为边数较小的子结构的过程称为减边变异。一个子结构可能有多个孩子,但加边变异只能从中选一个,当加边变异作了错误的选择时,减边变异便给了它重新选择的机会。例如,上文中 $S1$ 的加边变异有四个选择:$S(ac)$,$S(ab)$,$S(ae)$,$S(af)$,其中 $S(ab)$ 即 $S3$ 的适应值最大,但它并不是一个正确的选择,只有选择 $S(ac)$ 才有可能找到最优解 $S4$。可见没有减边变异,本次查找将以失败告终。

减边变异的过程是:设 Sub 为一个子结构,首先在 Sub 的历史中按适应值以轮盘赌方式选择 Sub 的一个祖先子结构记作 newSub,并作为结果返回;然后将 Sub 历史中所有 newSub 的祖先加入到 newSub 的历史中。

2.5 交叉

两个个体作交叉旨在将二者的优秀部分相结合以生成更为优秀的个体。个体带历史后,这一操作即成为可能,因为一个个体的历史中存放的就是该个体的若干优秀部分。因此子结构 Subst1 和 Subst2 的交叉过程可设计如下:1)按轮盘赌方式分别在 Subst1 和 Subst2 的历史中选择一个子结构 Sub1 和 Sub2;在 Sub1 的所有实例中任选一个实例,记作 ins1,在 Sub2 的实例中找出所有与 ins1 相重叠(有公共节点)的所有实例并任选其一作为 ins2,如没有相重叠的返回空值;2)将 ins1 和 ins2 合并成一个新的子图并转成一个子结构,记作 NewSub;3)收集 NewSub 的实例,将 Sub1 的任一实例和 Sub2 的任一实例合并成一个新的子图,在得到的这些子图中所有与 NewSub 同构的子图即为其实例;4)返回 NewSub。Sub1 和 Sub2 的历史信息也要相应地加入到 NewSub 的历史中。例如,$S(acf)$ 和 $S(abf)$ 交叉,在 $S(acf)$ 的历史中选 $S2$,$S(abf)$ 的历史中选 $S3$,然后在 $S2$ 的实例中任选一个实例,假设为 $g(4,5,11)$,在 $S3$ 中和 $g(4,5,11)$ 重叠的实例为 $g(4,11,12)$;二者合并为 $g(4,5,11,12)$,转成子结构 $S(abcd)$;其实例必然可由 $S2$ 和 $S3$ 的实例合并得到,最后只找到 $g(4,5,11,12)$ 为其实例,$S(abcd)$ 作为结果返回。

2.6 算法的伪码描述

下面是基于 EA 的子结构发现算法,其中 *popsize* 为种群大小,*limit* 为最大迭代次数,*pc* 为交叉概率,*pm* 为变异概率,*pma* 为变异时加边变异的概率,易见算法中加边变异的真实概率为 $pm \cdot pma$,减边变异的真实概率为 $pm \cdot (1 - pma)$,n 指定了爬山算法的邻域大小,*hw* 为用户指定的历史宽度。函数 *flip*(x) 随机产生一个介于 0 和 1 之间的随机数,如该数小于 x 则返回真,否则返回假。算法迭代地将种群 $P(t)$ 经过交叉变异生成种群 $C(t)$,然后用本文提出的选择机制从 $P(t)$ 和 $C(t)$ 中选择生成 $P(t+1)$,当达到最大迭代次数时返回目前找到的最优解。

```
输入: popsize, limit, pc, pm, pma, n, hw 等参数, t = 0
初始化种群 P(t) 并计算每个染色体的适应值
while (t < limit)
i = 0
do {在种群 P(t) 中随机选择两个个体 ch1, ch2
   if(flip (pc) )
   ch1 = ch1 和 ch2 的交叉结果;
   if (flip (pm))
   { if (flip (pma))
       ch1 = ch1 加边变异后的结果
     else
```

```
        ch1 = ch1 减边变异后的结果
    }
    将 ch1 保存为种群 C(t) 中的第 i 个个体
    i ++
} while (i < popsize)
从 P(t) 和 C(t) 中选择个体生成 P(t + 1)
t ++
返回最优个体
```

2.7 基于个体年龄段的选择机制

带历史的个体和加边变异中贪婪性的引入,在提高算法搜索能力的同时也对种群的多样性提出了更高的要求。为此设计了新的选择机制,它包括两部分的内容:第一部分旨在提高种群组成的多样性,因为查找过程是子结构边数每次加 1 逐渐增长的过程,将个体的年龄定义为该子结构的边数。种群 $C(t)$ 为中间种群,由种群 $P(t)$ 经过交叉变异生成,在选择时,首先将二者合并成一个大种群并找到其中最小个体的年龄 min、最大个体的年龄 max 和最优个体的年龄 *opt*, 然后将种群分为[min,(min + *opt*)/2],((min + *opt*)/2, (*opt* + max)/2),[(*opt* + max)/2,max]三个年龄段,对每个年龄段采取联赛选择的方式选择一半进入下代种群 $P(t+1)$。这种选择机制只在相同年龄段的个体间产生竞争,可以避免年轻而有潜力的个体由于发育不足而被淘汰,也可避免富有经验的年老个体(历史中含有有价值的子结构)被过早淘汰。

第二部分旨在提高种群中个体生成的多样性。新个体是通过扩展生成的,将 1.3 中的第 2)步,即新实例插入列表的方式改为随机插入,明显地提高了解的质量。这是由于一个子结构的实例不能重叠,否则无法利用 MDL 进行压缩,这样一个实例在插入时就要看它与前面已插入的实例是否重叠,如重叠则舍去。SUBDUE 和文献[9]中的算法采用固定的插入方式,因而不容易找到最优解。

2.8 去掉种群中没有潜力的个体

当一个个体的实例数为 1 时,它已没有发展潜力,因为从压缩率定义可以看出它对输入图没有压缩。这意味着一个子结构的实例数必须大于等于 2 才有意义,从而隐含给出了一个子结构大小的上界,即一个子结构的节点数必须小于等于输入图的节点数的一半,否则它不可能有两个不同的实例。这使得查找空间缩小了一半。

对于没有潜力的个体按一定的概率做减边变异或用从开始时生成的单边子结构中随机选出的个体来替换,这样可以避免无效计算,另外还具有重新初始化的效果,对种群多样性的保持也有帮助。

3 实验结果及分析

以 EPSD 表示文献[9]中的算法,以 HESD 表示本文的算法。EPSD 是基于进化编程(Evolutionary Programming,EP)的子结构发现算法,它只用了加边变异,没有交叉和减边变异,也没有与爬山算法结合。文献[9]从 http://ailab.uta.edu/subdue/的数据集中选择了 10 个图(见表 2),每个图代表一个网站,其中节点表示网页,边表示超链接。

文献[9]中的实验数据显示,EPSD 在表 2 数据上的实验结果全面超过了 SUBDUE。本文用 HESD 重新做了文献[9]中的实验,参数设置为: *popsize* = 30, *limit* 在 W10 中取 400,其他情形均取 100, $pc = 0.3$, $pm = 1$, $pma = 0.8$, $n = 3$, $hw = 8$。实验结果如表 3 所示。可以看出,除了在 *W*1 情形查找结果相同外,在其他情形 HESD 都超过了 EPSD。

带状态回溯的个体和爬山算法的引入从总体上提高了 EA 的寻优能力,而基于年龄段的选择机制和随机化个体的生成方式大大提高了种群的多样性,随时去掉种群中没有潜力的个体将查找空间降低了一半。这些机制正是 HESD 的实验结果较 EPSD 优秀的原因所在。

表 2　实验数据描述

图	节点数	边数	图	节点数	边数
W1	8	30	W6	85	303
W2	31	43	W7	27	492
W3	19	159	W8	25	532
W4	94	106	W9	86	501
W5	22	314	W10	732	16725

表 3　EPSD 和 HESD 发现的子结构对比

图	算法	发现的子结构			压缩率
		节点数	边数	实例数	
W1	EPSD	44	55	22	0.815789
	HESD				0.815789
W2	EPSD	68	57	33	0.743243
	HESD				0.635134
W3	EPSD	39	331	52	0.893253
	HESD				0.786516
W4	EPSD	88	77	34	0.865000
	HESD				0.795001
W5	EPSD	311	546	52	0.919643
	HESD				0.836309
W6	EPSD	310	210	195	0.817010
	HESD				0.806699
W7	EPSD	36	921	64	0.895954
	HESD				0.851636
W8	EPSD	46	1630	54	0.865350
	HESD				0.813286
W9	EPSD	44	66	1617	0.771721
	HESD				0.756390
W10	EPSD	22	11	345346	0.960646
	HESD				0.960532

4 结语

针对图数据挖掘中普遍存在的子图同构问题,使用了 SUBDUE 提出的带实例的子结构的概念,提出了带状态回溯个体的概念,从而使减边变异和交叉算子的设计更为合理。将 EA 和爬山算法融合,极大地提高了算法的寻优能力。基于年龄段的选择机制和随机化个体的生成方式大大提高了种群的多样性,以初始单边子结构替换种群中没有潜力的个体在缩小查找空间的同时进一步加强了种群的多样性。以上这些机制都为算法的效率和结果提供了有益的帮助。将本文提出的算法应用于图数据概念学习是今后的研究目标。

参考文献:

[1] INOKUCHI A, WASHIO T, MOTODA H. An Apriori-based algorithm for mining frequent substructures from graph data[C]// PKDD'2000, LNCS 1910. New York: ACM Press, 2000: 13 - 23.

[2] KURAMOCHI M, KARYPIS G. An efficient algorithm for discovering frequent subgraphs[J]. IEEE Transactions on Knowledge and Data Engineering, 2004, 16(9): 1038 - 1051.

(下转第 1951 页)

的“干草堆里的针”函数。也就是除了一个随机选取的串外,对其他所有串,子函数的值是 1。而对这个串,随机选取值 $1+\varepsilon$(正针)或 $1-\varepsilon$(负针)。我们的实验取 $\varepsilon=0.1$,强性参数 $k=4$,有 $l-k+1$ 个子函数,第 i 个函数 f_i 的支持集是 $\{i-1,i\cdots,i+k-2\}$,所以,在 f_i 和 f_{i+1} 间有一个 $k-1=3$ 的重叠。结果这个测试函数可有很多最优点。一般地,不是所有正针子函数同时有针集。因此,最优点值在 $l-k+1$ 和 $(l-k+1)(1+\varepsilon)$ 之间的某处,并且几乎所有情况都是次优点比最优点少 ε。

如果乘幂因子足够大,则从任何最优点的概率可确定最优点的近似个数,所有最优点的概率是相等的。当从分解中取样时,很容易累积由抽样产生的点的概率。

作为一个实例,运行 $l=20$,乘幂因子为 100,有 1 424 个最优点,由分解决定的最优点概率为 7.01528×10^{-4}。注意,$1/1424=7.022471910111\times10^{-4}$,若乘幂因子为 10 000,最优点的概率为 $7.022471910009\times10^{-4}$。图 1 显示对测试函数运行不同长度串的函数评价次数。根据文献[6]所给出的确定 j 阶探测次数 N 的公式:

$$N\geqslant\begin{cases}\ln(1-\delta^{1/J})/\ln(1-2^{j-k}), & \text{如果 } j<k\\ 1, & \text{如果 } j=k\end{cases}\qquad(15)$$

这里 J 是 j 阶强性集的个数,这些集以 $l\binom{k}{j}$ 为界。取成功概率 δ 为 0.9999,如 $l=64$,$k=4$ 和 $j=2$,则 $N=52.7$。

图 1 是常数次数 $l^2\log l$ 在文献[2]里预测的函数评价次数的复杂性。图 1 经验性地验证了算法的联接探测步骤函数评价次数的时间复杂性为 $O(l^2\log l)$。

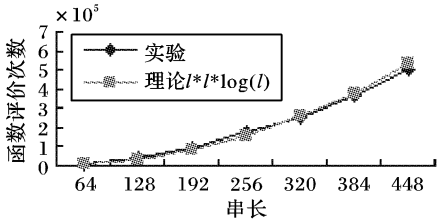


图 1　重叠随机针的函数评价次数

4　结语

本文先介绍了对一类在固定长度二元串上的适应度函数来说,从 Boltzmann 分布抽样是可行的。这类适应度函数需满足以下性质:1)它必须是 k- 强性的,即相互作用的变量(或比特)最多为 K 个。满足该条件的适应度函数有 NK-fitness,K 阶联接陷阱以及骗函数等。一个 k- 强性适应度函数是加性分解成一些依赖于最多 k bit 的子函数;2)加性分解子函数的支持集必须满足交叉性,即这些支持集有一个树形交叉性。

如果适应度函数是 k- 强性的,则本文算法的第一步(联接探测算法取自文献[1,2])从适应度函数中抽样确定 k- 强性结构。

这个随机的算法对任何事先给定的成功概率都获得成功。另外,算法计算函数的加性分解,如该加性分解满足交叉性,则进一步计算对任何给定的乘幂因子适应度函数的 Boltzmann 分布的分解。本算法的最后一步用这个分解从 Boltzmann 中抽样。如果选取足够大乘幂因子,算法将以高概率找到最优点。而且,算法还对最优点数量作准确的估计,所有最优点被等概率地抽样。

当 k 为常数时,本算法给出一个严格的复杂性分析,联接探测需要 $O(l^2\log l)$ 次函数评价,构建分解时间复杂性为 $O(l)$,从 Boltzmann 分布中抽样时间复杂性为 $O(l)$。

本算法的实现克服了在乘幂因子很大时可能出现的浮点溢出问题。给出了在一个函数的最优点和次优点适应度分离时选取乘幂因子的公式以及最优点与次优点的数量比。实验证明本算法是可行的。

参考文献:

[1] HECKENDORN R E, WRIGHT A H. Efficient linkage discovery by limited probing[C]// ERICK CANTU P A Z. Genetic and Evolutionary Computation (GECCO 2003), LNCS 2724. [S.l]: Springer-Verlag, 2003: 1003 - 1014.

[2] HECKENDORN R E, WRIGHT A H. Efficient linkage discovery by limited probing[J]. Evolutionary Computation, 2004, 12(4): 517 - 545.

[3] MUHLENBEIN H, MAHNIG T, RODRIGUEZ A O. Schemata, distributions and graphical models in evolutionary optimization[J]. Journal of Heuristics, 1999, 5(2): 215 - 247.

[4] MUHLENBEIN H, HONS R. The estimation of distributions and the maximum relative entropy principle[J]. Evolutionary Computation, 2005, 13(1): 1 - 28.

[5] MUHLENBEIN H, MAHNIG T. Convergence theory and application of the factorized distribution algorithm[J]. Journal of Computing and Information Technology, 1999, 7(1): 19 - 32.

[6] MUHLENBEIN H, MAHNIG T. FDA-a scalable evolutionary algorithm for the optimization of additively decomposed functions[J]. Evolutionary Computation, 1999, 7(4): 353 - 376.

(上接第 1947 页)

[3] KURAMOCHI M, KARYPIS G. Finding frequent patterns in a large sparse graph[J]. Data Mining And Knowledge Discovery, 2005, 11(3): 243 - 271.

[4] YAN X, HAN J W. gSpan: Graph-based substructure pattern mining[C]// Proceedings of IEEE International Conference on Data Mining (ICDM 2002), Maebashi City, Japan. Washington, DC: IEEE Press, 2002: 721 - 724.

[5] DEHASPE L, TOIVONEN H. Discovery of frequent datalog patterns [J]. Data Mining and Knowledge Discovery, 1999, 3(1): 7 - 36.

[6] RAEDT L D, KRAMER S. The levelwise version space algorithm and its application to molecular fragment finding[C]// Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01). San Francisco, CA: Morgan Kaufmann Pulishers, 2001, 2: 853 - 862.

[7] COOK D J, HOLDER L B. Graph-based data mining[J]. IEEE Intelligent Systems, 2000, 15(2): 32 - 41.

[8] GRUNWALD P. A tutorial introduction to the minimum description length principle[EB/OL]. [2006 - 12 - 26]. http://homepages.cwi.nl/pdg/ftp/mdlintro.pdf.

[9] BANDYOPADHYAY S, MAULIK U, COOK D J, *et al.* Enhancing structure discovery for data mining in graphical databases using evolutionary programming[C]// Proceedings of the Fifteenth International Florida Artificial Intelligence Research Society Conference. [S.l.]: AAAI Press, 2002: 232 - 236.

[10] HAN J W, KAMBER M. 数据挖掘概念与技术[M]. 2 版. 北京: 机械工业出版社, 2006.

[11] 李敏强, 寇纪淞, 林丹, 等. 遗传算法的基本理论与应用[M]. 北京: 科学出版社, 2002.