基于超完美图着色的存储分配算法*)

邓 宇 汪 黎 晏小波 王桂彬 唐 滔
(国防科技大学计算机学院并行与分布重点实验室 长沙 410073)

摘 要 为了提高性能,一些应用需要在编译时对主存进行针对性的管理。提出了基于超完美图的主存分配方法,其基本思想是通过生命周期分割将一般的相干图转换为超完美图,从而可以使用已有的线性时间的区间着色算法完成主存的分配。分别基于自底向上的积极生命周期分割策略和自顶向下的被动生命周期分割策略,实现了两个分配算法。初步评测表明,我们的分配算法是有效的编译时管理主存手段。

关键词 编译时存储分配,主存管理,区间着色,超完美图

Super-perfect Graph Coloring Based Memory Allocation Algorithms

DENG Yu WANG Li YAN Xiao-bo WANG Gui-bin
(PDL,School of Computer Science,National University of Defense Technology,Changsha 410073,China)

Abstract In some special applications,the main memory needs to be managed by the compiler to improve the performance. Memory allocation algorithms based on super-perfect graph coloring are proposed. The basic idea is to turn the general interference graph into super-perfect graph by live-range splitting and then use an existing linear time interval coloring algorithm to allocate the memory. Two allocation algorithms are implemented according to different live-range splitting strategies:a top-down aggressive strategy and a bottom-up passive strategy. Preliminary modeling tests show that they are efficient approaches for compile-time memory allocation.

Keywords Compile-time memory allocation,Main memory management,Interval coloring,Super-perfect graph

1 前言

在过去的十几年中,处理器速度以每年50%至100%平稳增长,而存储器的速度却只以每年7%左右增长[1]。处理器与存储器之间的速度差距导致了"存储墙"的出现[2],使得存储系统成为影响整个计算机系统性能的瓶颈,因此对存储系统的有效管理具有重要意义。

主存是存储系统的重要组成部分,其管理一般由操作系统完成。在一些实时处理领域和涉及核外计算(Out-of-Core Computation)[3]的大数据规模科学计算领域,计算机系统往往只运行一个固定的程序,用户也只关心单个程序的性能。在这些应用领域,操作系统的存储管理策略没有考虑程序特有的访存行为特征,因此其效率很低。而编译器可以获取有用的全局存储访问信息,能够系统地对主存进行管理,还可以减轻程序员的负担和减少运行时的开销。因此,对于特定的应用领域,从编译的角度来对主存进行管理是最好的选择。

在编译时进行存储分配的一个主要方法是对分配对象的生命周期相干图进行区间着色[4]。但一般图的区间着色问题过于复杂,至今没有线性时间的分配算法。利用程序的一些特殊性质可以化简区间着色的难度。本文利用程序中数组生命周期的包含性将一般图的区间着色问题归结为超完美图[4]的区间着色问题,而后者存在线性时间的最优解。为了将一般的相干图转换并保持为超完美图,必须对数组的生命周期进行分割。本文提出了两个分割数组生命周期的策略:一个是自底向上的积极分割策略,它首先将数组的生命周期分割到最小,然后在满足着色条件的前提下逐步合并生命周期;另一个是自顶向下的被动分割策略,它在一开始将数组的生命周期保持为最长,只有在不满足着色条件时才选择某些生命周期进行分割。基于这两个分割策略,我们实现了两个编译时的主存分配算法,并进行了模拟实验。结果表明,我们的算法是一种有效的编译时管理主存手段。

本文其余部分组织如下。第2节讨论了相关工作。第3节给出了区间着色的基本思想。第4节详细描述了基于不同生命周期分割策略的分配算法的实现,随后第5节给出了初步的模拟测试结果。最后对全文进行了总结。

2 相关工作

Fabri在1979年首先将区间着色与编译时存储分配联系在一起[5]。区间着色已被证明是一个NP完全问题[6],对于任意图进行区间着色非常困难。在Fabri之后,使用区间着色解决存储分配问题的工作主要集中在如何根据图的特殊性质来获得近似算法。Kierstead等人[7]研究了直线(straight-line)程序(即没有分支和循环的程序),将编译时存储分配问题建模为区间图[4]的区间着色问题,因为直线程序的相干图是区间图。但区间图的区间着色问题仍然是NP完全问题[6]。大部分真实程序都含有分支和循环,因此将相干图假设为区间图过于严格。许多研究者试图突破这个限制,寻找其它形式的图的区间着色。Confessore[8]以及Pemmaraju[9]研究了弦图[10]的区间着色问题。此外,研究人员试图通过一些程序相关的性质将相干图抽象为一些特殊的图。例如,Andersson[11]测试了大量程序的相干图,发现大部分相干图都是弦图。Li等人[12]利用数组生命周期的特殊性质将相干

*)国家自然科学基金60621003和60633050。邓 宇 博士研究生,研究方向为计算机体系结构和编译等。

图抽象为超完美图，而对超完美图进行区间着色存在线性时间的最优解。但 Li 的工作只限于对便签存储器(scratchpad memory)的分配，不能用于对主存的分配。此外，Li 提出的策略没有给出当相干图不能构成超完美图时的处理方法。

3 区间着色的基本思想

编译时的主存分配问题可以描述如下：给定一个大小为 M 的用户可以使用的主存空间，以及待分配空间的数组 A_1，A_2，…，A_n，其大小分别是 $L(A_1)$，$L(A_2)$，…，$L(A_n)$，我们的任务是：(1)为每个数组在主存中分配空间；(2)产生主存与外存之间的数据传输语句(即 IO 函数调用)。

一般而言，所有数组大小的总和 $L(A_1)+L(A_2)+\cdots+L(A_n)$ 大于 M。我们假设对于每个循环，计算所需要的数据能够全部放入主存，否则可以通过循环分块减小参与一次循环的数组大小。因此，编译器要决定在什么地方插入主存与外存之间的数据传输语句，使得在计算开始之前所需要的数据已经位于主存。分配数组时要尽可能把经常使用的数组保留在主存中，并且要将生命周期重叠的数组分配到不重叠的空间，以免造成存储器的抖动，增加 IO 操作。数据传输语句可以用于分割数组的生命周期，改变同一个数组在主存中的位置，因此这两个任务是互相联系的。

通过对数组进行生命周期分析，可以构造数组生命周期的相干图，图中每个结点表示待分配空间的数组，结点之间的边表示这两个数组的生命周期重叠。主存分配问题可以归结为对相干图进行区间着色的问题。区间着色问题的定义如下：

给定一个带权图 $G=(V,E)$ 和一个结点权值映射函数 $w:V\rightarrow N$，N 是自然数集。区间着色的问题是为 V 中每个结点 u 指派一个区间 I_u，使之满足以下条件：(1)对任意 $u\in V$，$|I_u|=w(u)$；(2)对任意 $(u,v)\in E$，$I_u\cap I_v=\phi$。所谓区间就是指数轴上的一段闭区间。

与带权图相关的概念还包括团(clique)、色数(chromatic number)以及团数(clique number)等。团是完全子图的结点集合，有时候也指代完全子图。如果一个团不被图中任何其它团所包含，这个团就称为极大团。一个团的阶是团中所有结点的权值之和，而团数是图中所有极大团的阶的最大值。色数是对图进行区间着色所需的区间跨度总和 $|\bigcup_u I_u|$ 的最小值。

区间着色的目标是获得刚好达到色数的一种区间指派。一般而言一个带权图的色数大于等于其团数。在特殊情况下，如果色数等于团数，这样的图被称为超完美图。超完美图存在线性时间的区间着色算法。

一个数组的生命周期是包含该数组在程序控制流图上的所有活跃结点的闭区间。数组在控制流图的某个结点上活跃是指该数组的某个元素在这个结点上被读或者修改。两个数组的生命周期可能存在分离、相交和包含 3 种关系，如图 1 所示。分离是指没有一个控制流图的结点同时属于两个数组的生命周期，相交是指存在控制流图的结点同时属于两个数组的生命周期，而包含则是指某个数组的生命周期所覆盖的所有控制流图结点都属于另一个数组的生命周期。

Li 等人[12]证明，如果程序中所有数组的生命周期之间都不存在相交的情况，也就是它们要么分离，要么相互包含，那么由此得到的数组生命周期的相干图是超完美图，因此可以使用一个线性时间的区间着色算法完成主存的分配。

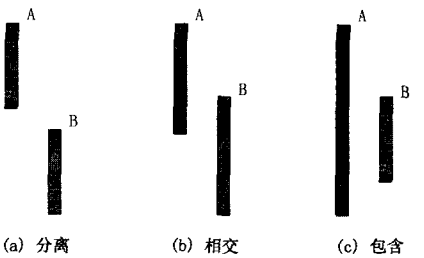


图 1 生命周期之间的关系

然而并非所有程序的数组生命周期都满足这样的性质。当数组的生命周期之间存在相交的情况时，可以通过插入显式的数据传输语句，分割或者合并生命周期，消除相交的情况，使相干图成为超完美图。采用不同的生命周期分割策略，我们提出了两个基于区间着色的主存分配算法。

4 分配算法实现

分割生命周期的目的有两个：一个是当相干图不可着色(即用户可使用的主存空间 M 小于相干图的团数)时，必须分割数组的生命周期，使部分数组产生溢出，减小着色压力，使相干图满足着色条件。这是采用图着色分配算法进行资源分配时必须采取的一个步骤。另一个目的是将相干图转换并保持为超完美图。我们采取了两种分割生命周期的策略：一种是积极的分割策略，它首先将各个数组的生命周期分割到最小，然后在满足着色条件和保持超完美图的前提下逐渐将一些数组的生命周期合并起来，从而减少与外存的数据传输次数。另一种是被动的分割策略，它使得每个数组的生命周期尽可能长，只有到万不得已的时候才进行分割。基于这两种分割策略，我们实现了两种主存空间的分配算法，分别称为基于积极分割策略的分配算法(简称积极分配算法)和基于被动分割策略的分配算法(简称被动分配算法)。下面讨论它们的实现。

4.1 基于积极分割策略的分配算法

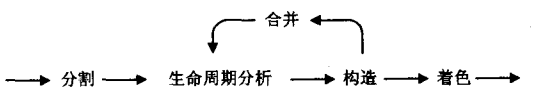


图 2 积极分配算法过程示意图

图 2 给出了积极分配算法的过程，各个步骤完成的工作如下。

分割：在分割步骤，编译器在每个循环前面插入对数组的加载语句，在循环后插入对数组的回存语句，这样所有数组的生命周期都被分割到最小，即限制在一个循环内。

生命周期分析：将控制流图上的一个循环块作为基本单位，对数组进行生命周期分析。

构造：首先根据生命周期分析的结果构造相干图，同时计算出相干图的团数和每个数组的生命周期，然后检查数组生命周期是否存在相交的情况。如果当前相干图的团数大于给定的主存大小，或者数组生命周期存在相交的情况，则回退到上一次合并操作前的相干图。

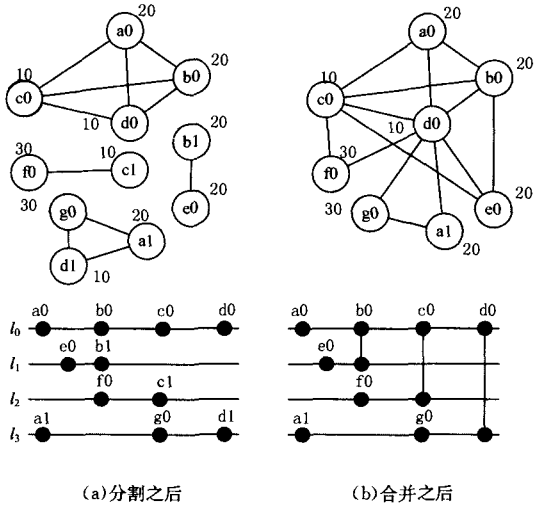
合并：对数组生命周期进行合并，合并之后要重新进行生命周期分析并构造相干图。由于合并可能破坏着色条件，或者使数组生命周期出现相交的情况，因此每次构造相干图后都要进行检查，如果不满足条件，则放弃这次合并。

着色：当生命周期无可合并时，使用一个线性时间的区间着色算法[12]完成主存的分配。

图3给出了一段将用于说明分配算法的程序代码。程序声明了7个整型数组，方括号里是数组的大小。程序由4个循环组成，在循环中，数组名出现在等号的右边表示对该数组元素的引用，出现在左边表示对该数组元素的赋值。

```
// 数组定义
1     int a[20], b[20], c[10],
      d[10], e[20], f[30], g[30];

// 程序体
2     for (...){
3           ... = a;
4           ... = b;
5           c = ...;
6           d = ...;
7     }

8     for (...){
9           ... = b;
10          e = ...;
11    }

12    for (...){
13          ... = c;
14          f = ...;
15    }

16    for (...){
17          ... = a;
18          ... = d;
19          g = ...;
20    }
```

图3　用于说明分配算法的程序代码



(a)分割之后　　(b)合并之后

图4　积极分配算法下不同阶段的相干图和生命周期示意图

假设主存大小为60。图4给出了不同阶段的相干图(上半部分)和数组生命周期示意图(下半部分)。相干图中的结点表示图3中的数组，旁边的数字表示数组大小。下半部分的生命周期示意图中的4条横线表示4个循环。完成分割之后的相干图如图4(a)上半部所示，分割时对数组进行了重命名，$a0$和$a1$都表示数组a，其余以此类推。分割之后的相干图由4个相互独立的团构成，每一个团与一个循环对应。每个数组的生命周期都是相应循环上的一个点。此时相干图中阶最大的极大团是$\{a0,b0,c0,d0\}$，其阶为60，刚好等于主存大小，因此可以着色。下面进行生命周期的合并，以使更多的数组能够在其整个生命周期内都保存在主存中，减少IO传输次数。在依次合并$\{b0,b1\}$、$\{c0,c1\}$和$\{d0,d1\}$之后，得到如图4(b)所示的相干图和生命周期示意图。这时不能再对$\{a0,a1\}$进行合并，因为合并之后相干图中将出现阶为80的极大团$\{a0,b0,c0,d0,e0\}$，导致相干图不可着色。从图4(b)下半部分的生命周期示意图可见，没有任何两个数组的生命周期有相交的情况。因此图4(b)就是最终用于着色的相干图。

最后，按照数组生命周期包含关系的非递减序来进行着色。所谓数组生命周期包含关系的非递减序，是指这样一个序列，在这个序列中，如果数组a排在数组b之前，那么a的生命周期要么与b分离，要么包含于b。例如，$\{a0,e0,f0,a1,g0,b0,c0,d0\}$就是这样一个序列。首先对$a0$着色，它被放置在偏移地址为0的地方。然后是$e0$，它与已着色的数组$a0$不相干，因此也被放置在偏移地址为0的地方。数组$f0$和$a1$同样如此。数组$g0$与$a1$相干，因此被放置在偏移地址为20的地方。同理依次完成对$b0$，$c0$和$d0$的着色。最终各数组在主存中的分配结果如图5所示。

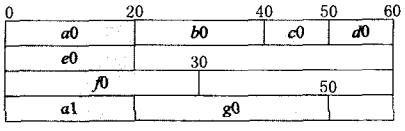


图5　积极分配算法的分配结果

4.2　基于被动分割策略的分配算法

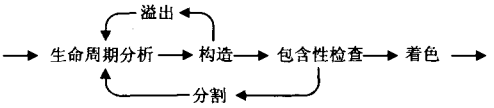


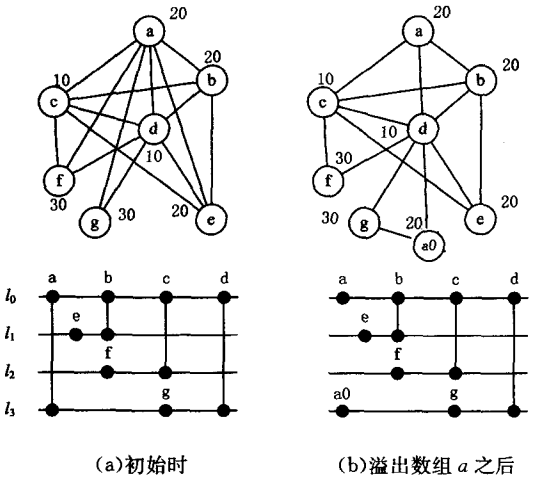
图6　被动分配算法过程示意图

图6给出了被动分配算法的过程，它由两个循环构成，第一个循环是“生命周期分析→构造→溢出”，其目的是检查相干图是否满足着色条件，即是否存在阶大于主存大小的极大团。如果存在，就通过溢出步骤来分割其中部分数组的生命周期(相当于从极大团中去掉这些数组结点)，减小极大团的阶，直到所有的极大团的阶都小于主存大小。

第二个循环是“包含性检查→分割→生命周期分析→构造”，其目的是保证相干图是超完美图。最开始数组的生命周期可能存在相交的情况，而且随后进行溢出也可能导致相交的出现。因此，当相干图的着色条件得到满足后，还要进行包含性的检查。如果两个数组的生命周期相交，可以通过延长或者分割其中一个数组的生命周期，使它们不相交。由于延长生命周期可能增加相干图的团数，导致相干图不可着色，而分割生命周期则不会出现这样的情况。因此我们采取了分割生命周期这种保守的策略。

最后进行着色时采用的方法与积极分配算法相同。

对于图3给出的程序代码，图7(a)给出第一次生命周期分析后构造的相干图和生命周期示意图。同样假设主存大小为60。由于相干图中存在一个阶为80的极大团$\{a,b,c,d,e\}$，因此要通过溢出步骤对这个极大团中的部分数组进行生命周期的分割。这里我们选择数组a，因为它在程序中没有被修改过，为这样的数组生成溢出代码时不需要生成回存语句。将a溢出后的相干图和生命周期示意图如图7(b)所示，a的生命周期被分割为两部分，分别是循环l_0和l_3上的一个点。此时相干图满足着色条件，得到的分配结果与图5相同。



(a)初始时　　(b)溢出数组 a 之后

图 7　被动分配算法下不同阶段的相干图和生命周期示意图

5　评测

我们采用了一个模拟与分析相结合的方法来评测第 4 节所提出的分配算法的性能。评测方法由以下 3 个步骤组成：

第一步，通过分配算法获得数组在主存中的分配方案。我们在 GCC 中实现了两种生命周期分割策略下的分配算法，能够自动对程序进行分析并生成分配方案。

第二步，通过 profiling 获得程序的访存地址序列。

第三步，分析访存地址序列，统计给定主存分配方案下访问外存的 IO 开销，作为评估分配算法性能的指标。

为了进行比较，我们实现了一个简单的在线分配算法。该算法采用类似首次适应(First Fit)分配算法[13]的思想，从当前可用空间的最低地址开始为新出现的数组分配空间。如果没有可用空间，则将最近最少使用的数组替换出去。我们称这个分配算法为参考算法。

我们选取了 NPB 中的 MG，SPEC2000 中的 Swim，Jacobi(完成 Jacobi 迭代算法的程序)以及 GEMM(二维矩阵乘法)来进行评测，测试程序的规模如表 1 所示，所有程序都按照各自规模进行了分块。在固定的数据规模下，我们改变主存大小，考察了各个分配算法的编译开销和性能。

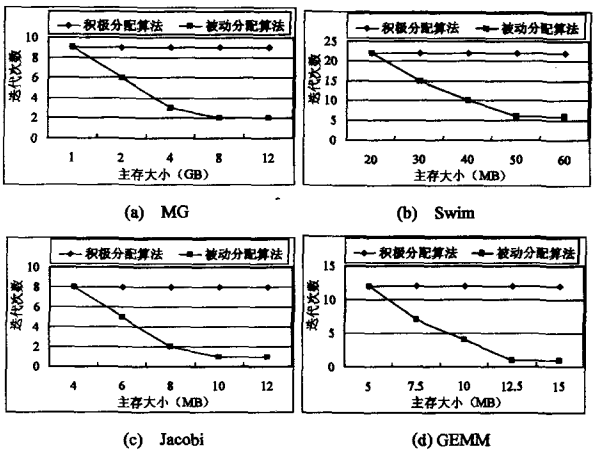
表 1　评测主存分配算法的测试程序

程序	来源	数组数目 *	规模(双精度浮点)
MG	NPB	15	512×512×512
Swim	Spec2000	28	1024×1024
Jacobi	-	12	1024×1024
GEMM	-	15	1024×1024

* 分块后的数组单独计算

首先考察基于积极分割策略和被动分割策略的两个分配算法的编译开销。这两个算法都需要经过若干次迭代才能得到满足要求的分配方案，前者通过迭代不断合并生命周期，后者通过迭代不断分割生命周期，因此我们以迭代的次数作为衡量分配算法的编译开销的指标。

图 8 给出了对于不同的测试程序，两种分配算法的编译开销随主存大小变化的曲线图。从图中可以看出，积极分配算法的迭代次数不随主存大小变化而变化，而被动分配算法的迭代次数则随主存的增大而减小。这是因为前者总是尝试对所有可能合并的生命周期进行合并，不管能否合并成功。而后者只在必要时才分割生命周期，随着主存增大，着色压力减小，产生的溢出随之减少，因此迭代次数也减少。当主存继续增大到超过色数时，被动分配算法的迭代次数不再减小，这些迭代是为满足包含性而进行的生命周期分割，与主存大小无关。

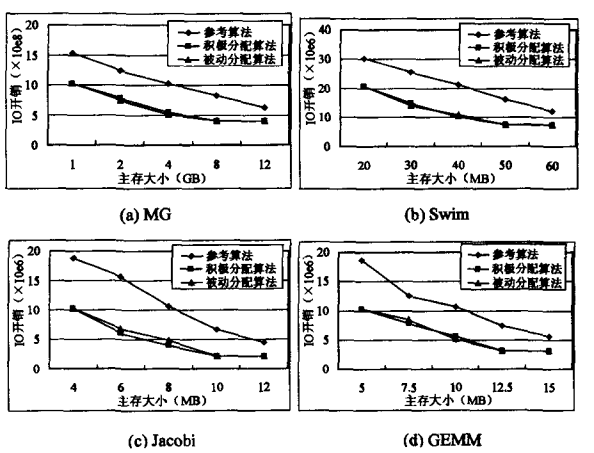


(a) MG　　(b) Swim

(c) Jacobi　　(d) GEMM

图 8　积极分配算法与被动分配算法的编译开销比较

然后考察参考算法、积极分配算法和被动分配算法的性能。图 9 给出了对这三种分配算法的评测结果。可见，随着主存的增大，所有算法产生的开销都在减小。积极分配算法与被动分配算法效果接近，它们产生的开销要比参考算法少很多。



(a) MG　　(b) Swim

(c) Jacobi　　(d) GEMM

图 9　不同分配算法产生的 IO 开销比较

结束语　区间着色是编译时常用的主存分配方法。本文提出了基于超完美图的主存分配方法，其基本思想是通过生命周期分割将一般的相干图转换为超完美图，从而可以使用已有的线性时间的区间着色算法完成主存的分配。根据不同的生命周期分割策略，实现了两个分配算法。一个基于自底向上的积极分割策略，首先将数组的生命周期分割到最小，然后在满足着色条件的前提下逐步合并生命周期。另一个基于自顶向下的被动分割策略，它在一开始将数组的生命周期保持为最长，只有在不满足着色条件时才选择某些生命周期进行分割。初步评测表明，我们的分配算法是一种有效的编译时管理主存手段。

参 考 文 献

[1]　Patterson D A, Hennessy J L. Computer Architecture: A Quantitative Approach. Second Edition. China Machine Press, 1999

(下转第 257 页)

(2)迭代向后逐步删除归约。其算法思想,基本类同于算法1,故略。

(3)决策树遗传归纳。在每个节点选取"最好"的属性,并将其分类。

算法2 决策树遗传归纳算法

输入:初始属性集合$\{A_1A_2\cdots A_i\cdots A_n\}$
输出:归约属性集合 A
算法开始:
①置 k=0,随机产生属性效用值的初始种群 $\vec{X}(0)$:
$\vec{X}(0)=(X_1(0),\cdots,X_N(0))\in\{A_1A_2\cdots A_i\cdots A_n\}$;
②独立地从当前种群中选取 N-1 个母体;
③独立地从所选取的 N-1 个母体进行边际效用值杂交,得到 N-1 个中间母体;
④独立地对杂交后的 N-1 个个体进行 $\frac{U'(x_i)}{\sum_{i=1}^{N}U'(x_i)}$ 变异,得到 k+1 代种群的前 N-1 个个体:
$X_1(k+1),\cdots,X_{N-1}(k-1)$;
⑤计算 $i=\arg\max\{U'(X_j(k))\}$,令 $X_N(k+1)=X_i$,则停止;
⑥若不满足,则 k=k+1 并返回②;
算法结束。

3.2 新属性的学习

新属性的学习主要包括属性初始命名学习、属性归纳与分析命名。

(1)属性初始命名学习

若语句结构体的语义值为真,从取值集合中按照多值对应的原则。如果属性库中无相应的属性记录,则确定其属性并命名其名称,按照概念的成分、性质与事实来确定属性。例如,假设服装类库中没有颜色的属性,但是在叙述中存在说"服装是雪色",就是说雪的属性值是白色,其属性名称命名为"颜色",并且设定该属性的效用初值为1。

(2)属性的归纳与分析命名学习

通过数据为属性的训练样例进行属性泛化,利用一个边际效用属性函数,得到新属性保持原有的数据一致性。在属性归约中,常对已知数据进行小波变换、主成分分析、线性变换、对数变换、桶变换(直方图等)、聚类(距离变换)等,基本算逻辑运算的函数变换,本算法中我们用属性效用的投影、积、差、极值、最值、级数边际效用法对不同的对象进行不同的表达式学习,进而命名新属性,进行属性的数据类型有界限定,从而得到新属性。

算法3 属性效用的边际效用涣化算法

输入:初始属性集合$\{A_1A_2\cdots A_i\cdots A_n\}$
输出:新命名属性
算法开始:
①FOR i=1 TO n;
② FOR j=1 TO i;
③ 利用所有 A_j 的边际效用值,施行投影、积、差、极值、最值、级数变换;
④ 如果变换后保持原来的数据一致性,则命名新的属性;
⑤ ENDFOR j;
⑥ ENDFOR i;
⑦构造新属性的数据类型,得到新的属性;
算法结束。

据此,便可设计基于效用的结构语法属性学习模型系统。例如,在某服装厂服装属性学习系统实验设计中,便可采用服装作为所论对象来进行其概念的属性学习,使其输入文本是一段该服装厂关于客户需求的说明,通过服装的客户需求,分解单词,确定所有的 β_i 和 α_i,并对每个 α_i,β_i 进行属性学习。从而通过对属性的自动学习,来获得其服装客户的各属性(包括姓名、性别、电话、接件日期、取件日期、上体长、手臂长、胸围、颈围、腰围、臀围、肩宽、胸宽、背宽、前腰节高、后腰节高、总体高、身高、下体长)。

结束语 在属性的学习中,把属性当成数据进行分析和归纳,目前有许多问题尚待解决。例如,学习环境的描述及其设计问题;对于已知环境为随机事件并且已知其分布,可采用基于统计的机器学习法;结构语法文本分析;文本单词分解;等等。但由于其复杂性,本文仅对结构语法的文本进行了初步分析和一定研究,并假定文本是结构语法描述的,且为简单的结构语法,故仅为引玉之砖。进一步的研究,至少应有:如果文本不是简单的结构语法描述,而是复杂的结构语法描述,则需要对文本的单词用复杂的结构语法进行有效分解,才能得到 α_i 和 β_i;学习的表达式构造方法绝不止用属性的效用和边际效用来构造,故相关算法也应进行深入的分析、研究和设计;数据的数据类型和范围尚需更详细的数据类型描述工具(显然,如果数据类型的描述分类更小,则理当为属性的自动学习找到更有效的方法。比如,顾客数据包括年龄和年薪属性,要根据实际情况进行属性的规范化处理,年薪的取值范围可能比年龄的取值范围要大得多)。此外,在实际应用系统的设计中,还可用中间件、组件等技术思想,对不同类施行不同的属性自动学习。

参考文献

[1] Han Jiawei, Kamber M. Data Mining: Concepts and Techniques. Second Edtion. 机械工业出版社,2006
[2] 杨祥茂,等. 基于网络资源消费者模型的调度策略. 计算机科学,2003,30(9):105-106
[3] 杨祥茂,等. 基于均衡模型的计算机资源分配. 计算机科学,2005,32(4):171-172
[4] 刘颖. 计算机语言学. 清华大学出版社,2002
[5] Michell T M. 机器学习. 机械工业出版社,2003

(上接第236页)

[2] Wulf W A, McKee S A. Hitting the Memory Wall: Implications of the Obvious. Computer Architecture News, 1995, 23(1): 20-24
[3] Bordawekar R, Choudhary A, Kennedy K, et al. A Model and Compilation Strategy for Out-of-core Data Parallel Programs// Proceedings of ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. ACM Press, 1995: 1-10
[4] Golumbic M C. Algorithmic Graph Theory and Perfect Graphs. A-nnals of Discrete Mathematics, 2004
[5] Fabri J. Automatic Storage Optimization// SIGPLAN '79: Proceedings of the SIGPLAN Symposium on Compiler construction. 1979: 83-91
[6] Johnson M R. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., 1979
[7] Kierstead H A. The Linearity of First-fit Coloring of Interval Graphs. SIAM J. Discrete Math 1, 1988: 526-530
[8] Confessore G, Dell'Olmo P, Giordani S. An approximation result for the interval coloring problem on claw-free chordal graphs. Discrete Appl. Math., 2002, 120(1-3): 73-90
[9] Pemmaraju S V, Penumatcha S, Raman R. Approximating interval coloring and max-coloring in chordal graphs. J. Exp. Algorithmics, 2005, 10: 2-8
[10] West D B. 图论导引. 原书第2版. 李建中,骆吉洲,译. 机械工业出版社,2006
[11] Andersson C. Register Allocation by Optimal Graph Coloring// CC'03: Proceedings of the 12th International Conference on Compiler Construction. Springer-Verlag, 2003
[12] Li L, Nguyen Q H, Xue J. Scratchpad Allocation for Data Aggregates in Supperperfect Graphs. ACM SIGPLAN NOT., 2007, 42(7): 207-216
[13] Tanenbaum A S, Woodhull A S. 操作系统:设计与实现(第二版). 王鹏,尤晋元,朱鹏,熬青云,译. 电子工业出版社,1998