算法框架:算法重定位的一种可操作的方法*

栾尚敏[1] 李 未[1] 马绍汉[2]

[1](北京航空航天大学计算机科学与工程系 北京 100083)

[2](山东大学计算机科学系 济南 250100)

摘要 算法框架是实现算法重定位的一种可操作的方法.文章给出了算法重定位的定义,讨论了如何通过选择操作和组合操作来设计算法框架.然后给出了定义选择操作和组合操作的一种方法.由文章所定义的算法框架得到求解某一问题的算法模式,从该算法模式就可以生成求解该问题的算法.文章最后讨论了算法框架和算法模式以及模块化算法之间的关系.研究算法重定位及其可操作的方法对软件自动化和机器学习有着重要的意义.

关键词 算法设计,软件自动化,算法重定位.

中图法分类号 TP301

算法设计自动化是软件自动化的难点和关键.算法设计是一种知识高度密集的创造性活动,要实现算法设计自动化,难度很大.在算法自动或半自动生成方面虽然取得了一定的进展[1~4],但是,这些方法只能生成具有特定结构的算法,例如循环算法和递归算法,或只能生成特定领域的算法,而目前还没有一种生成通用算法的方法.

Lü[1]在半自动生成算法的系统NDADAS(ND algorithm design automation system)中采用了问题归约的方法,用这种方法可以生成具有特定结构的算法,例如循环算法和递归算法等.Helman[2]给出了搜索问题的一种描述方法和该类问题的一个算法模式,讨论了如何通过该算法模式得到求解具体问题的动态规划和分枝限界算法.我们进一步推广了Helman的方法,给出了求解问题的一个算法模式[3].通过该算法模式,我们不仅可以得到动态规划和分枝限界算法,而且可以得到贪心和回溯算法.而后,我们[4]又给出了一个算法模式来得到求解问题的混合算法,但采用这种方法来实现算法的自动设计只能限制在一定领域内.

本文从算法重定位的角度来讨论如何实现算法自动化.首先给出了算法重定位的定义,讨论了算法框架的设计,以及如何由策略控制关系和变换来定义横向组合操作和纵向组合操作.由算法框架可以得到求解某一问题的算法模式,再由这个算法模式得到求解该问题的算法,所以,文献[2~4]中所定义的算法模式可以由本文所给出的算法框架得到.本文最后讨论了算法框架和模块化算法[5]之间的关系.算法框架是算法重定位的一种可操作的方法,研究算法重定位及其可操作的方法对实现算法设计的自动化和机器学习有着重要的意义.

本文以优化代码问题为例进行讨论.设有 n 个元素 $a_1,a_2,\ldots,a_n$,每一个 a_i 对应着一个数值,记为 $d(a_i)$,$1\leqslant i\leqslant n$,试构造一个二叉树,这个二叉树以这 n 个元素为叶节点,使 $\sum_{i=1}^{n}L_i*d(a_i)$ 最小.这里,L_i 是从根节点到标记为 a_i 的叶节点的路径长度.

1 算法重定位的定义和算法框架的设计

在讨论算法重定位和算法框架之前,我们首先讨论问题的结构和问题的实例,也就是说如何来描述一个问

* 本文研究得到国家自然科学基金资助.作者栾尚敏,1968年生,博士,讲师,主要研究领域为算法设计自动化,形式化方法,人工智能.李未,1943年生,博士,教授,博士生导师,中国科学院院士,主要研究领域为并发程序设计语言,操作语义,类型理论,人工智能.马绍汉,1938年生,教授,博士生导师,主要研究领域为算法,并行算法,人工智能.

本文通讯联系人:栾尚敏,北京 100083,北京航空航天大学计算机科学与工程系

本文1998-02-23收到原稿,1998-08-27收到修改稿

题及其实例.需要说明的是,本文所给出的算法框架并不依赖于这里所给出的问题的描述方法的,只是为了方便叙述和讨论.

定义 1.1(问题的结构). 用二元组 $P=(A,Q)$来描述问题的结构,这里,A 是原子的集合.用 MS 表示求解过程中产生的所有中间结果的集合,它可以是二叉树的集合,也可以是一个串的集合,或是一个集合的集合等等.Q 是解的集合.

定义 1.2(问题的实例). 用四元组(P,I,∞,d)来描述问题的实例.这里,P 是问题的结构,如上所述;I 是初始状态的集合;∞是一个 Q 上的偏序关系;d 是一个辅助集合,在某些问题中它用来存放描述问题的数据,例如货郎担问题和优化代码问题.在某些问题中也用来存放其他内容,例如,在 SAT 问题中它是子句的集合.

下面,我们以优化代码问题为例来说明如何用这种描述问题的方法描述具体问题.

$$A=\{a_1,a_2,\ldots,a_n\}$$

$$Q=\{t\mid t \text{ 是以 } a_1,a_2,\ldots,a_n \text{ 为叶节点的二叉树}\}$$

$$I=\{a_1,a_2,\ldots,a_n\}$$

用 d 来存放 $a_1,a_2,\ldots,a_n$ 的值.为了定义偏序关系∞,还应定义 MS 上的一个函数 f 如下:

设 $A'\subset A$,t 是以 A' 中的元素为叶节点的二叉树,则

$$f(t)=\sum_{a\in A'} L*d(a).$$

设 $t_1,t_2\in Q$,若 $f(t_1)\leqslant f(t_2)$,则 $t_1\infty t_2$.

定义 1.3(算法重定位). 对于求解某一具体问题的算法,我们可以对它进行修改,从而改变求解问题的方法,或者使得修改后的算法成为求解另一个问题的算法,对算法的这种修改称为算法重定位.

从算法重定位的定义可以看到,对于求解某一问题的算法,我们可以对它进行修改,使求解问题的方法发生了改变,但该算法仍然是求解原问题的算法,这种修改是算法重定位.对于一个算法,我们可以通过对它的修改,使它成为求解另一问题的算法.不论求解问题的方法是否改变,这种修改也是算法重定位.

直观上讲,对于一个算法,我们只要修改它的某些部分,就可以使求解问题的方法发生改变,或者使它成为求解另一个问题的算法.例如,在下面的过程中

Step1. $MS=I$.

Step2. 若不存在 t_1 使得 $R_1(t_1)$成立,则停机.否则从 MS 中选择一个二叉树 t_1,使得 $R_1(t_1)$成立.

Step3. S 是 MS 的一个子集,并且 S 中的任意一个二叉树 t_2,$R_2(t_2)$成立.

Step4. S 中的每一个二叉树 t_2,构造一个二叉树 t,使得 t_1 和 t_2 分别是 t 的左儿子和右儿子,$MS=MS+\{t\}$.

Step5. GOTO Step 2.

设 I 是上面所述的代码优化问题中所定义的初始状态的集合.若以如下方式定义 R_1 和 R_2:

若 t_1 不是 MS 中其他任何树的子树,并且满足这种条件的树中 t_1 的代价最小,则 $R_1(t_1)$成立;

若 t_2 不是 MS 中其他任何树的子树,并且满足这种条件的树中 t_2 的代价次小,则 $R_2(t_2)$成立,

就得到代码优化问题的贪心算法.

若以如下方式定义 R_1 和 R_2:

若在 MS 中存在树 t,t_1 和 t 没有公共的元素作为叶节点,分别以 t 和 t_1 作为左儿子和右儿子的树不存在于 MS 中,并且 t_1 的代价比 MS 中任何一个满足如下条件的树 t_3 的代价小:t_3 以 A 中的 n 个元素作为叶节点,则 $R_1(t_1)$成立;

对于 $t_2\in MS$,若 t_2 的代价比 MS 中任何一个满足如下条件的树 t_3 的代价小:t_3 以 A 中的 n 个元素作为叶节点,则 $R_2(t_2)$成立,

就得到代码优化问题的分枝-限界算法.

如果有一种方法能描述出各种算法的这些不同的地方,那么要修改一个算法就成为一件很容易做到的事情.如何来描述这些不同的地方呢?这将在下文加以叙述.

在上面的过程中,Step 2 是从 MS 中选择一个满足一定条件的元素 t_1,我们称这一过程为选择操作.Step 3

是从 MS 中选择一个满足一定条件的元素 t_2，Step 4 是执行变换 $tr(t_1,t_2)$；我们把执行 Step 3 和 Step 4 的过程称为组合操作. 下面给出选择操作和组合操作的准确定义.

设 A 是一个集合，用 2^A 表示集合 A 的幂集，即集合 A 的所有子集组成的集合.

定义 1.4(选择操作). 称部分函数 $OP:2^{MS}\to MS$ 是一个选择操作，如果 OP 满足：$\forall\ x\in 2^{MS}, OP(x)\in x$. 所有的选择操作组成的集合记为 SS. 这里，MS 是求解过程中产生的中间结果的集合.

定义 1.5(组合操作). 称部分函数 $COP:MS\times 2^{MS}\to 2^{MS}$ 是一个组合操作，所有的组合操作组成的集合记为 CS. 这里，MS 是在求解过程中所产生的中间结果的集合.

下面我们给出基于选择操作和组合操作的一个算法框架.

```
alframework(▽)
  begin
    MS=I;
    while (not complete(MS)) do
      begin
        select1(SS,▽);
        select2(CS,▽);
      end
  end
end(alframework)
```

这里，▽是一个集合，根据▽中的内容决定采用哪种选择操作和组合操作. 下面我们将详细说明如何通过该算法框架得到求解具体问题的算法.

2　定义组合操作和选择操作的一种方法

在第1节我们曾经指出，若能对各种算法的不同之处给出一种准确的描述，就可以很容易地实现算法的重定位. 下面将讨论这个问题.

定义2.1(选择控制关系). 设 # 是 MS 上的一个二元关系，且满足对于任意的 $t_1,t_2\in Q, t_1\#t_2$ 当且仅当 $t_1\propto t_2$，则称 # 为关于 $\propto$ 的选择控制关系.

定义2.2(组合控制关系). 设 $<$ 是 MS 上的一个二元关系，且满足对于任意的 $t_1,t_2\in Q, t_1\propto t_2$ 成立，则 $t_2\#t_1$ 一定不成立，称 # 为关于 $\propto$ 的组合控制关系.

定义2.3(变换). 若部分函数 tr 是 $MS\times MS\to MS$ 的一个函数，则称这个部分函数 tr 为变换. 用 tr 或其下标的形式来表示.

定义2.4(策略控制关系). 设 #，$<$ 分别是关于 $\propto$ 的选择控制关系和组合控制关系，则称序对 $(\#,<)$ 为策略控制关系.

下面以货郎担问题为例来说明如何定义一个具体问题的某种求解方法的策略控制关系和变换. 关于货郎担问题的详细描述请参见文献[6]. 要详细地描述货郎担问题的分枝-限界方法，需定义如下策略控制关系和变换.

选择控制关系 #：设 $t_1\in MS, t_2\in Q$，若 t_2 的边权之和小于 t_1 的边权之和，则 $t_2\#t_1$.

组合控制关系 $<$：设 $t_1\in MS, t_2\in Q$，若 t_2 的边权之和小于 t_1 的边权之和，则 $t_2\#t_1$.

变换 $tr(t_1,t_2)$：将路径之和合并为一个路径.

货郎担问题的分枝-限界算法的选择控制关系和组合控制关系是相同的，但在很多情况下，它们也是不同的，例如该问题的动态规划算法. 这里还有两个问题需要说明. (1) 对于某些具体问题的算法，有的并不非要定义其选择控制关系和组合控制关系不可，如 SAT 问题的 GSAT[7] 算法，只定义其组合控制关系就行了，因为其当前解很明显，就是最近得到的 n 维向量. (2) 组合控制关系有时是依赖于当前解的，这个关系在不同的时刻可能不同. 下面给出的这两个选择操作和组合操作说明了如何用策略控制关系来定义选择操作和组合操作.

```
COP1(t,MS)
  begin
    T=∅
    while (S3(MS,Δ,TS,T,t)≠null)
      begin
        ⟨t',tr⟩=S3(MS,Δ,TS,T,t);
        T=T+{t'}
        while (S4(MS,Δ,T,t')≠null)
          begin
            ⟨t1,≺⟩=S4(MS,Δ,T,t');
            if (t1≺t') then
              t'=t1;
            T=T+{t1};
          end
        t1=tr(t,t');
        MS=M+{t1};
      end
    return MS
end (COP1)
```

```
OP1(MS)
  begin
    t=S1(MS,Λ,d);
    while (S2(MS,Λ,t,B,d)≠null) do
      begin
        ⟨t',#⟩=S2(MS,Λ,t,B,d);
        if (t'#t) then
          t=t'
        B=B+{t'}
      end
    return t;
  end (OP1)}
```

```
COP2(t,MS)
  begin
    while (S3(MS,Δ,TS,T,t)≠null)
      begin
        ⟨t',tr⟩=S3(MS,Δ,TS,T,t);
        t1=tr(t,t')
        MS=MS+t1;
      end
    return MS
  end (COP2)
```

```
OP2(MS)
  begin
    t=S1(MS,Λ,d);
    return t;
  end(OP2)
```

根据定义可以很容易地证明如下性质.

性质2.1. OP_1和OP_2是选择操作.

性质2.2. COP_1和COP_2是组合操作.

并不是所有的算法都可以通过这里定义的两个选择操作和两个组合操作而得到,但可以定义一些新的选择操作和组合操作得到所需要的算法.在定义一种选择操作或组合操作时,应尽量使所定义的选择操作或组合操作适用的情况多一些,而不是只适用于一种情况.由这里定义的两个选择操作和两个组合操作可以得到4种算法模式.

3 算法重定位的实现

本节讨论如何由上一节的算法框架来实现算法重定位,也就是说,如何通过上述的算法框架来得到求解具体问题的算法.若选择操作和组合操作如第2节所定义,则算法框架中的变量▽中的内容为策略控制关系.根据▽中的策略控制关系确定选用哪种选择操作和组合操作,这样也就确定了一种算法模式,再对算法模式中的变量和未定义的函数给出定义,就得到了求解具体问题的算法.例如,要得到货郎担问题的分枝限界算法,根据我们定义的分枝-限界方法的选择控制函数和组合控制函数,需要选择$OP_1(MS)$和$COP_1(MS)$来作为选择操作和组合操作,可得到如下的算法模式.

```
algorithm HL
  begin
    MS=I;
    while (not complete(MS)) do
      begin
        t=OP1(MS);
        MS=COP1(t,MS);
      end
  end(algorithm)
```

若 A 和 Δ 中分别包含第2节中定义的货郎担问题的分枝-限界方法的选择控制关系和组合控制关系，$complete(MS)$的定义同文献[4]给出的定义，再定义该算法模式中的变量 $S_1(MS,A,d)$;$S_2(MS,\Delta,t,B,d)$;$S_3(MS,\Delta,TS,T,t)$;$S_4(MS,\Delta,T,t')$就得到求解货郎担问题的分枝-限界算法. 定义如下.

$S_1(MS,A,d)$:从 MS 中任意选择一个元素.

$S_2(MS,A,t,B,d)$:从 MS 和 A 中分别选择一个元素和一个选择控制关系，直到已取遍了 MS 和 A 中的所有元素时为 $null$.

$S_3(MS,\Delta,TS,T,t)$:从 MS 中任意选择一个元素，且不存在一个 A 中的原子，该原子同时存在于上述元素和 t 中，直到取遍了 MS 中满足上述条件的路径时为 $null$;并且也选择一个变换作为返回值的一部分.

$S_4(MS,\Delta,T,t')$:从 MS 和 A 中分别选择一个元素和一个选择控制关系，直到已取遍了 MS 和 A 中的所有元素时为 $null$.

只要选取选择操作 OP_2和组合操作 COP_1，再给出策略控制关系的适当定义就可以得到求解 SAT 问题的 GSAT[7]算法了，并且通过算法框架也可以得到混合算法. 限于篇幅，不再详细讨论. 通过上面的例子可以看到如何通过算法框架来实现算法重定位.

4 讨　论

前面给出了算法重定位的定义，讨论了如何由选择操作和组合操作来设计算法框架，以及如何由算法框架得到求解具体问题的算法. 这里首先说明如何用面向对象的方法编程实现. 首先建立一个类，该类对应着算法框架，然后对于不同的选择操作和组合操作，定义不同的子类，这样就得到了不同的算法模式. 对于算法模式的编程实现问题，在文献[3,4]中都给予了讨论，文献[6]还用 C++实现了一个系统. 限于篇幅，这里不再多述，有兴趣的读者请参见文献[3,4,6].

由上面的结果我们可以看到，文献[2～4]中所给出的算法模式可以由本文所给出的算法框架得到. 我们还推广了文献[2～4]中的结果. 例如，本文所定义的选择操作就是对文献[2,3]中所定义的横向组合运算的推广，也就是说横向组合运算是选择操作中的一种. 通过我们所定义的组合选择控制关系和变换可以很容易地来描述组合操作的过程，而文献[2～4]却没有做到这一点. 另外，文献[2～4]所给出的算法模式是基于问题的一种严格的描述方法，而这里我们所给出的算法框架并不是基于某一种问题的描述方法，虽然我们也给出了问题结构和问题实例的一种描述，但这只是为了叙述和讨论的方便，我们还可以给出其他形式的描述方法，设计基于该种问题描述的选择操作和组合操作就可以了.

一个算法，若它的各种应用的不同是附加在这个算法的前面或后面，而没有必要再改写原来的算法，则称该算法是模块化算法[5]. 在用面向对象的方法进行程序设计时，我们若已经对一个模块化算法进行了编码，当我们再用到该算法时就可以建立一个子类，将该算法继承下来使用，而没有必要再对它重新进行编码. 对于我们这里给出的算法框架，在用面向对象的方法进行程序设计时，也有同样的优点. 例如，我们在给出了一个选择操作或组合操作的编码后，当我们再次用到该选择操作或组合操作时，就没有必要对它进行重新编码，而直接继承下来使用就可以了.

参考文献

1 Lü Jian. Framework of algorithm correctness in NDADAS. Science in China (series A), 1991,34(7):875～884

2 Helman P. An algebra for search problems and their solution. In: Kanal L, Kumar V eds. Search in Artificial Intelligence. Berlin: Springer-Verlag, 1988. 28～90

3 栾尚敏,马绍汉.一类问题的描述方式及其算法.计算机学报,1995,18(10):755～762
(Luan Shang-min, Ma Shao-han. A common model for a class of problems and their algorithms. Chinese Journal of Computers, 1995,18(10):755～762)

4 栾尚敏,马绍汉.搜索问题的代数描述及其算法.计算机研究与发展,1997,34(11):801～806
(Luan Shang-min, Ma Shao-han. An algebraic model for seach problems and their algorithm. Computer Research and Development, 1997,34(11):801～806)

5 Tello E R. Objected-oriented Programming for Artificial Intellegence. Berlin: Springer-Verlag, 1989

6 栾尚敏.面向对象的方法库设计[硕士学位论文].济南:山东大学,1993
(Luan Shang-min. Object-oriented method base design [MS. Thesis]. Ji'nan: Shandong University, 1993)

7 Selman B, Levesque H, Mitchell D. A new method for solving hard satisfiability problem. In: Clancey W ed. Proceedings of the 10th National Conference on Artificial Intelligence (AAAI-92). San Jose, CA: MIT Press, 1992. 440～446

Algorithm Framework: an Operational Approach to Algorithm Relocation

LUAN Shang-min[1] LI Wei[1] MA Shao-han[2]

[1](*Department of Computer Science and Engineering Beijing University of Aeronautics and Astronautics Beijing* 100083)

[2](*Department of Computer Science Shandong University Ji'nan* 250100)

Abstract Algorithm framework is an operational approach to algorithm relocation. In this paper, the authors present the definition of algorithm relocation and an algorithm framework based on selection operator and combination operator. The authors discuss how to design selection operator and combination operator. The algorithm for a given problem can be obtained from the algorithm schema, which is obtained from the algorithm framework presented in this paper. The relationship between algorithm framework and algorithm schema and the relationship between algorithm framework and modular algorithm are discussed as well. Algorithm relocation and its operational approach are significant for software automation and machine-learning.

Key words Algorithm design, software automation, algorithm relocation.