软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2011,22(9):1985−1993 [doi: 10.3724/SP.J.1001.2011.03948]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

组合优化问题简约与算法推演*

郑宇军[1,2+], 薛锦云[1,2], 凌海风[3]

[1](中国科学院 软件研究所 计算机科学国家重点实验室,北京 100190)

[2](江西师范大学 江西省高性能计算重点实验室,江西 南昌 330027)

[3](南京大学 管理工程学院,江苏 南京 210093)

Combinatorial Optimization Problem Reduction and Algorithm Derivation

ZHENG Yu-Jun[1,2+], XUE Jin-Yun[1,2], LING Hai-Feng[3]

[1](The State Key Laboratory of Computer Science, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[2](Provincial Key Laboratory of High Performance Computing, Jiangxi Normal University, Nanchang 330027, China)

[3](School of Management and Engineering, Nanjing University, Nanjing 210093, China)

+ Corresponding author: E-mail: yujun.zheng@computer.org

Zheng YJ, Xue JY, Ling HF. Combinatorial optimization problem reduction and algorithm derivation. *Journal of Software*, 2011,22(9):1985−1993. http://www.jos.org.cn/1000-9825/3948.htm

Abstract: A unified algebraic model is used to represent optimization problems, which uses a transformational approach that starts from an initial problem specification and reduces it into sub-problems with less complexity. The model then constructs the problem reduction graph (PRG) describing the recurrence relations between the problem, and derives an algorithm with its correctness proof hand-in-hand. A prototype system that implements the formal algorithm development process mechanically is also designed. This approach significantly improves the automation of algorithmic program design and helps to understand inherent characteristics of the algorithms.

Key words: combinatorial optimization problem; problem reduction; algorithm derivation; PAR (partition-and-recur); correctness proof

摘 要: 针对组合优化类问题定义了代数结构模型,从问题的形式规约出发,通过一阶谓词和量词演算将问题逐步简约为搜索空间更小、复杂度更低的子问题,根据问题的简约关系推导出求解算法,并在构造算法的同时也证明了算法的正确性.开发了原型系统以支持上述形式化的开发过程.这种算法推演技术能够显著提高算法程序设计的自动化水平,而问题简约的思想也更有利于对算法本质特征的理解.

关键词: 组合优化问题;问题简约;算法推演;PAR(partition-and-recur);正确性证明

中图法分类号: TP301 **文献标识码**: A

组合优化问题是指在离散有限的数学结构中和给定的约束条件下寻找目标函数最优值的问题,它是组合数学、运筹学和理论计算机科学中长期研究的重要问题之一.传统的组合优化算法设计策略(如动态规划、贪心、回溯等)有着各自不同的适用范围,而且缺乏策略选择的有效标准[1],因而极大地限制了算法开发的形式化

* 基金项目: 国家自然科学基金(61105073, 60773054); 科技部国际科学技术合作项目(2008DFA11940)
收稿时间: 2009-09-25; 定稿时间: 2010-09-06

和自动化水平.

20世纪80年代以来,人们提出了一些抽象代数模型来精确刻画组合优化问题的结构和求解过程,但复杂的表示法使其难以在计算机上进行有效实现.如 Helman[2]的搜索模型将解空间视为一组连接子的集合,求解过程就是通过反复执行连接子的横向组合和纵向组合来对解空间进行隐式枚举搜索,组合过程中采用的不同计算辅助关系可以用于不同类型的算法设计[3,4].Kumar 和 Kanal 等人[5−7]对与或图、游戏树等特定结构上的最优化问题进行了形式化的定义,通过抽象的分支限界和动态规划方法对这些结构上的多种具体算法进行了统一归纳,从而简化了算法的开发和证明,但其应用范围在很大程度上受到数据结构的限制,一些搜索算法还要求特殊的结构性质,适应性和可扩展性不强.Bird 和 Meertens 提出的 B-M 公理表示法[8,9]同样提倡从问题规约出发,通过应用融合(fusion)和提升(promotion)等一系列定理逐步推导算法,但该方法需要在各种问题结构上定义大量的定理和规则,其表示法不易于使用[10],且推导结果很难直接转换为结构化程序.

除了递归程序变换等较简单的功能,早期的算法生成系统[11−13]需要设计人员的干预较多.南京大学开发的 NDADAS 系统[14,15]基于函数分解树模型来将问题逐步分解成可直接求解的若干子函数,通过 CASE 结构形成、直接替换、递归节点生成等规则来支持许多算法的机械化设计,但并不保证这些规则和分解出的子函数能够满足求解需求,其后续研究也转向使用机器学习等技术简化人工设计[16].美国 Kestrel 研究所的 KIDS 系统[17]则通过演绎推理、有限差分、部分求值和类型求精等一系列技术生成算法程序,后续的 Designware 系统[18]进一步将设计知识分类存放在数据库中,并借助范畴论工具来提高自动化水平.不过,其最关键的设计策略在很大程度上仍需要手工选择,问题/算法的种类和范围也始终存在较大的限制.

以我们前期的 PAR(partition-and-recur)方法[1,19]研究为基础,同时借鉴相关研究中算法形式化开发和程序自动生成的思想,本文提出了一种新的组合优化算法设计方法.它从组合优化问题的代数规约出发,通过谓词和量词演算将问题逐步简约为更易求解的子问题,在此基础上得到正确高效的算法程序;开发的实验系统已成功应用于多个典型组合优化算法的自动生成.

1 问题简约模型和方法

使用结构化的代数规约,一个问题可由问题输入域以及从输入到输出的构造方式来刻画.给定问题输入域 D 和输出域 R,那么一般组合优化问题的规约可定义为

$$P(d)\equiv(\mathbf{Q}z{:}c(z)\wedge z\in\zeta(d){:}f(z)) \tag{1}$$

其中:$\mathbf{Q}$ 为二元运算符 $\mathbf{q}$ 的一般化量词,对于优化类问题,$\mathbf{q}$ 一般取 min 或 max;$\zeta{:}D\to SET(Z)$为解空间生成函数;$c{:}Z\to BOOLEAN$ 为解的可行性约束函数;$f{:}Z\to REAL$ 为解的目标函数.

人们处理复杂问题的一个基本方法就是将其分解为更简单的子问题.采用穷举法求解问题(1)时,需要搜索的解空间为$\{z|c(z)\wedge z\in\zeta(d)\}$,那么我们对同一结构上问题间的关系作如下定义[20]:

定义 1. 问题 $P'(d')=\langle\zeta',c',f\rangle$称为问题 $P(d)=\langle\zeta,c,f\rangle$的等价问题,当且仅当$\{z|c'(z)\wedge z\in\zeta'(d')\}=\{z|c(z)\wedge z\in\zeta(d)\}$.

定义 2. 问题 $P'(d')=\langle\zeta,c',f\rangle$称为问题 $P(d)=\langle\zeta,c,f\rangle$的子问题,当且仅当$\zeta'(d')\subseteq\zeta(d)$并且$\forall z\in\zeta(d'){:}c'(z)\Rightarrow c(z)$.

定义 2 还可以细分为以下 3 种情况:

定义 3. 如果 $P'(d')=\langle\zeta,c',f\rangle$为 $P(d)=\langle\zeta,c,f\rangle$子问题,且 $c'\neq c$,那么称 $P'(d')=\langle\zeta,c',f\rangle$为 $P(d)$的派生子问题.

定义 4. 问题 $P(d')=\langle\zeta,c,f\rangle$称为问题 $P(d)=\langle\zeta,c,f\rangle$的直接子问题,当且仅当$\zeta(d')\subseteq\zeta(d)$.

定义 5. 问题 $P'(d')=\langle\zeta,c\wedge c_1,f\rangle$称为问题 $P(d)=\langle\zeta,c,f\rangle$的强化约束子问题(简称为强化问题);相反,$P(d)$称为 $P'(d')$的放松约束问题(简称为放松问题).

将函数分解关系代入初始问题规约,通过谓词和量词演算来对规约进行变换,不断地将问题简约为更易求解的子问题,最后根据简约过程和演算结果得出问题的求解算法.

问题的简约过程可以通过问题简约图(problem reduction graph,简称 PRG)形象地描述.PRG 是一个有向无环图,其中每一个节点都代表一个问题;图中的一个无入边的特殊节点称为根节点,它表示要求解的原始问题;无出边的节点称为叶节点或终点,它们表示可直接求解的问题;设问题 $P(d)$被简约为一组子问题 $P_1(d_1)$,

$P_2(d_2),\ldots,P_n(d_n)$,那么从 $P(d)$到每个 $P_i(d_i)$绘制一条边,$P_i(d_i)$称为 $P(d)$的子节点$(0<i\leqslant n)$.

2 算法推演技术

2.1 基本演算规则

从问题的初始代数规约出发,通过不断应用数学定律、一阶谓词演算、量词性质等演算规则,能够对问题规约进行不断变化,从而进行问题简约和算法推演.除了常用的数学定律和谓词演算规则外,表 1 中的量词性质在组合优化算法推演中起到关键性的作用.

Table 1 Main quantifier rules for algorithm calculation

表 1 主要的量词演算规则

Rule	Equations	Remark
Cartesian product	$(\mathbf{Q}i,J\colon r(i)\wedge s(i,J)\colon f(i,J)) \equiv (\mathbf{Q}i\colon r(i)\colon (\mathbf{Q}J\colon s(i,J)\colon f(i,J)))$	J is a non-empty set of any constrained variable, and $s(i,J)$ is a predicate of i and J
Range splitting	$(\mathbf{Q}i\colon r(i)\colon f(i)) \equiv (\mathbf{Q}i\colon r(i)\wedge b(i)\colon f(i))\ \mathbf{q}\ (\mathbf{Q}i\colon r(i)\wedge\neg b(i)\colon f(i))$	$b(i)$ is a boolean expression of i
Single range	$(\mathbf{Q}i\colon i=k\colon f(i)) \equiv f(k)$	
Range disjunction	$(\mathbf{Q}i\colon r(i)\vee s(i)\colon f(i)) \equiv (\mathbf{Q}i\colon r(i)\colon f(i))\ \mathbf{q}\ (\mathbf{Q}i\colon s(i)\colon f(i))$	**q** is an idempotent operator
Functional associativity	$(\mathbf{Q}i\colon r(i)\colon f(i)\ \mathbf{Q}\ g(i)) \equiv (\mathbf{Q}i\colon r(i)\colon f(i))\ \mathbf{q}\ (\mathbf{Q}i\colon r(i)\colon g(i))$	
Functional commutativity	$(\mathbf{Q}i\colon r(i)\colon (\mathbf{Q}j\colon s(j)\colon f(i,j))) \equiv (\mathbf{Q}j\colon s(j)\colon (\mathbf{Q}i\colon r(i)\colon f(i,j)))$	
General distributivity	$(\mathbf{Q}i\colon r(i)\colon g\otimes f(i)) \equiv g\otimes(\mathbf{Q}i\colon r(i)\colon f(i))$	$\otimes$ is a commutative binary operator and is associative with respect to q, and x is not a free variable of g
Functional distributivity	$g((\mathbf{Q}i\colon r(i)\colon f(i))) = (\mathbf{P}i\colon r(i)\colon g(f(i)))$	$g(a\ \mathbf{q}\ b) = g(a)\ \mathbf{p}g(b)$ where **P** is the generalized quantifiers of **p**
$\forall$-rule	$(\forall i\colon r(i)\wedge s(i)\colon p(i)) \equiv (\forall i\colon r(i)\colon s(i) \Rightarrow p(i))$	
$\exists$-rule	$(\exists i\colon r(i)\wedge s(i)\colon p(i)) \equiv (\exists i\colon r(i)\colon s(i)\wedge p(i))$	
Conditional separation	$(\mathbf{Q}i\colon r(i)\vee(s(i)\wedge t(i))\colon f(i)) \equiv \begin{cases}(\mathbf{Q}i\colon r(i)\colon f(i))\ \mathbf{q}\ (\mathbf{Q}i\colon s(i)\colon f(i)), & \text{if } t(i)\\ (\mathbf{Q}i\colon r(i)\colon f(i)), & \text{else}\end{cases}$	

基于抽象规约(1),对组合优化问题进行简约的出发点一般有两种:

(1) 分解生成函数$\zeta(d)=\zeta_1(d)\cup\zeta_2(d)\cup\ldots\cup\zeta_m(d)$;

(2) 分解约束函数 $c(z) = c_1(z)\cup c_2(z)\cup\ldots\cup c_m(z)$.

将函数分解关系代入初始问题规约,之后就可以通过谓词和量词演算来对规约进行变换,不断将问题简约为更易求解的子问题,最后根据简约过程和演算结果得出问题的求解算法.

2.2 算法推演与效率提升

毋庸置疑,抽象规约(1)本身就蕴含了问题的求解算法,即遍历解空间$\zeta(d)$中每一个满足可行性约束 $c(z)$的解,找出其中目标函数值 $f(z)$最优的一个.但这样的穷举算法在绝大多数情况下都是低效的,各种高效算法则往往需要设计人员去发现和证明.而使用逻辑推演来简约问题,能够自动确定问题与子问题最优解之间的关系,同时,自动发现并合并相同或等价的子问题(即在 PRG 中不能包含两个相同或等价的问题节点),从而提高了算法的求解效率.

例 1:最大子段和问题.给定一个整数序列 $A_n=\{a_1,a_2,\ldots,a_n\}$,要找出其元素之和最大的子序列.设 ss 为所有子序列的生成函数,那么该问题的规约为

$$MS(A_n)\equiv(\mathbf{MAX}z\colon z\in ss(A_n)\colon sum(z)) \tag{2}$$

该问题的生成函数可以划分为 $ss(A_n)=\{a_n\}\cup ss(A_{n-1})\cup\{z|z=z'\uparrow a_n\wedge z'\in ss(A_{n-1})\wedge last(z')=a_{n-1}\}$,即 A_n 的子序列要么是 A_{n-1} 的子序列,要么以 a_n 结尾(可以只含 a_n).将其代入问题规约并进行推演可得:

$MS(A_n)$

≡(**MAX**z:$z=a_n\vee z\in ss(A_{n-1})\vee(z=z'\uparrow a_n\wedge z'\in ss(A_{n-1})\wedge last(z')=a_{n-1}$:$sum(z)$)

≡[范围分裂]

max((**MAX**z:$z=a_n$:$sum(z)$),(**MAX**z:$z\in ss(A_{n-1})$:$sum(z)$),(**MAX**z':$z'\in ss(A_{n-1})\wedge last(z')=a_{n-1}$:$sum(z'\uparrow a_n)$))

≡[单点范围和卷叠]

max(a_n,$MS(A_{n-1})$,(**MAX**z':$z'\in ss(A_{n-1})\wedge last(z')=a_{n-1}$:$sum(z')+a_n$))

≡[令 $MS'(A_n)\equiv$(**MAX**z:$z\in ss(A_n)\wedge last(z)=a_n$:$sum(z)$)]

max(a_n,$MS(A_{n-1})$,$MS'(A_{n-1})+a_n$)

≡[$a_n\leqslant MS'(A_{n-1})+a_n$]

max($MS(A_{n-1})$,$MS'(A_{n-1})+a_n$)

其中,$MS'(A_n)$为原问题的派生子问题,对其继续进行推演可得:

$MS'(A_n)$

≡[等价问题]

(**MAX**i:$0<i\leqslant n$:(Σj:$i\leqslant j\leqslant n$:a_j))

≡[范围分裂:$(0<i\leqslant n)\equiv(0<i\leqslant n-1)\vee(i=n)$]

max((**MAX**i:$0<i\leqslant n-1$:(Σj:$i\leqslant j\leqslant n$:a_j)),(**MAX**i:$i=n$:(Σj:$i\leqslant j\leqslant n$:a_j))

≡[单点范围和卷叠]

max($MS'(A_{n-1})+a_n$,a_n))

将原问题与子问题的简约关系合并,就可以得到如下线性算法:

Algorithm ***MS***($\boldsymbol{A_n}$).

```
begin
  ms←0; ms′←0;
  for i=1 to n do
    ms′←max(ms′+a_i,a_i);
    ms←max(ms,ms′);
  return ms;
end.
```

例 2:最小时间调度问题.给定要在某机器上完成的一组任务 $S_n=\{1,2,\ldots,n\}$,其中,第 i 项任务需要的处理时间为 t_i.要确定这些任务的一个调度,使得所有任务的完成时间之和最小.设 z 为 S_n 的一个排列,那么其中第 z_i 项任务的完成时间就是(Σj:$0<j\leqslant i$:t_{z_j}).令 ps 返回一个序列的所有排列,那么该问题的规约

$$MTS(S_n)\equiv(\mathbf{MIN}z:z\in ps(S_n):(\Sigma i:0<i\leqslant n:(\Sigma j:0<j\leqslant i:t_{z_j})))\tag{3}$$

该问题的生成函数可以划分为 $ps(S_n)=(\cup i:0<i\leqslant n:s_i\uparrow ps(S_n\backslash\{s_i\}))$,将其代入问题规约并进行推演可得:

$MTS(S_n)$

≡(**MIN**z:$z\in(\cup k:0<k\leqslant n:k\uparrow perms(S_n\backslash\{k\}))$:($\Sigma i$:$0<i\leqslant n$:$(n+1-i)\,t_{z_i}$))

≡[交叉积]

(**MIN**k:$0<k\leqslant n$:($\Diamond z$:$z\in perms(S_n\backslash\{k\})$:$nt_k+(\Sigma i:0<i\leqslant n-1:(n-i)\,t_{z_i})$))

≡[范围分裂和单点范围]

(**MIN**k:$0<k\leqslant n$:$nt_k+(\Diamond z:z\in perms(S_n\backslash\{k\}):(\Sigma i:0<i\leqslant n-1:(n-i)\,t_{z_i}))$)

≡[卷叠]

(**MIN**k:$0<k\leqslant n$:$nt_k+MST(S_n\backslash\{k\})$)

这样得到的是原问题与其多个子问题的简约关系,其蕴含的算法效率并不高.不过,对此还可以做进一步简约:易证 $MST(S_n\backslash\{i\})-MST(S_n\backslash\{j\})\leqslant n(t_j-t_i)$,故 $i\leqslant j\Rightarrow nt_i+MST(S_n\backslash\{i\})\leqslant nt_j+MST(S_n\backslash\{j\})$,那么令 $k^*=$(**MIN**k:$0<k\leqslant n$:t_k),简约关系中其他子问题就都可以消去,最后得到:

$$MTS(S_n)\equiv nt_{k^*}+MST(S_n\backslash\{k^*\}) \tag{4}$$

从简约关系(4)中很容易得到求解该问题的线性算法,即按 t_k 由小到大的顺序安排任务调度.

例 1 和例 2 推演得到的都是线性算法,但它们既有相同点又有不同点.相同点是二者都存在最优子结构性质,这是动态规划法的基本特征[21];基于最优结构的贪心算法则可视为是动态规划法的精化[22].例 1 推演得到的算法属于动态规划算法;而例 2 还满足特殊的性质,故进一步得到了贪心算法.

许多可应用贪心算法的问题都具有如下性质:在将原问题简约到多个子问题时,如果这些子问题满足某些附加的单调性质,使得只有一个或少数几个子问题需要保留,而其他的子问题可以被舍去.我们称此简约为"贪心简约".形式化地,设问题 $P(d)$与其子问题之间的简约关系为(以最小值问题为例)

$$P(d)\equiv\min(P_1(d\odot e_1)\oplus w(e_1),P_2(d\odot e_2)\oplus w(e_2),\ldots,P_n(d\odot e_n)\oplus w(e_n)) \tag{5}$$

其中,$e_1,e_2,\ldots,e_n\in E$ 称为问题的单点域,$\odot:D\times E\to D$ 称为单点函数,$\oplus$为目标函数值域上的二元算子.如果能够找到 E 上的某个单调函数 v,使得 $v(e_i)\leqslant v(e_j)\Rightarrow P_i(d\odot e_i)\oplus w(e_i)\leqslant P_j(d\odot e_j)\oplus w(e_j)$,那么可以选取各单点对象中 v 值最小的 e_k,这样就得到了对于 $P(d)$的贪心简约:

$$P(d)\equiv P_k(d\odot e_k)\oplus w(e_k) \tag{6}$$

这种方式不仅适用于推演符合拟阵结构[23]的贪心选择算法,也同样适用于不符合拟阵结构的其他贪心算法(如以上最小时间调度算法、Huffman 编码算法等).

2.3 分支限界、近似和参数化策略

动态规划和贪心法是寻求多项式时间算法的两种基本方法,难以有效应用对于大量 NP 难题.分支限界法是为复杂问题设计确定性算法的主要手段,通过问题简约过程来生成分支限界算法的基本步骤为:

(1) 将原始问题标记为活节点,其目标函数收益为 0;

(2) 选取一个活节点,对其进行简约得到一组子问题;

(3) 对于可直接求解的子问题,计算其目标函数值,并对当前最优解进行更新;

(4) 舍去那些无解或不能导致最优解的子问题,而将其余的子问题标记为活节点,其目标函数收益由父节点及其到当前节点的简约关系确定;

(5) 重复第(2)步,直至不再有未处理过的活节点.

设某个子问题的目标函数值上界为 x,而该节点的当前收益为 y,如果 $x+y$ 不优于当前最优解,那么可以判断其不能导致最优解,该节点可以被舍弃.问题的上界可由其放松问题的解来确定,而活节点的选取方式则决定了分支限界法是按照广度优先、深度优先还是效益优先的方式执行.

以 0-1 背包问题为例,给定背包容量 W 和一组物品 $A_n=\{a_1,a_2,\ldots,a_n\}$,其中,$a_i$ 的重量和价值分别为 $w(a_i)$和 $v(a_i)$,那么问题规约为

$$KS(A_n,W)\equiv(\mathbf{MAX}z:w(z)\leqslant W\wedge z\in P(A_n):v(z)) \tag{7}$$

其生成函数为幂集函数,分解形式为 $P(A_n)=P(A_{n-1})\cup\{z|z=z'\cup\{a_n\}\wedge z'\in P(A_{n-1})\}$,那么推演过程为:

$KS(A_n)$

$\equiv(\mathbf{MAX}z:w(z)\leqslant W\wedge(z\in P(A_{n-1})\vee(z=z'\cup\{a_n\}\wedge z'\in P(A_{n-1}))):v(z))$

≡[范围分裂]

$\max((\mathbf{MAX}z:w(z)\leqslant W\wedge(z\in P(A_{n-1}):v(z)),(\mathbf{MAX}z':w(z'\cup\{a_n\})\leqslant W\wedge z'\in P(A_{n-1}):v(z'\cup\{a_n\})))$

≡[卷叠]

$\max(KS(A_{n-1}),(\mathbf{MAX}z':w(z')\wedge z'\leqslant W-w(a_n)\wedge z'\in P(A_{n-1})=v(z')+v(a_n)))$

≡[条件分离和卷叠]

$$\begin{cases}\max(KS(A_{n-1},W),KS(A_{n-1},W-w(a_n))+v(a_n)), & \text{if } w(a_n)\leqslant W\\ (KS(A_{n-1},W), & \text{else}\end{cases}$$

问题的简约可一直持续到 $KS(\Phi,W')=0$,采用深度优先的策略就得到了问题的回溯算法.再通过上界限制能

够显著减小搜索范围,其上界可为无约束问题$\mathbf{MAX}z:z\in P(A_n):v(z)$的解$v(A_n)$,也可为放松约束的连续背包问题的解(由贪心算法求得).

许多 NP 难题在实际应用中只需要找到一个近似最优解.在算法推演的过程中,可以舍弃一些复杂的问题节点来提高算法效率,并根据问题的简约关系来估算解的近似程度.

仍以 0-1 背包问题为例,不失一般性,令 $w(a_i)\leqslant W$,那么在前述推演的基础上可得:

$$KS(A_n,W)\equiv(\mathbf{MAX}i:0<i\leqslant n:KS(A_n\backslash a_i,W-w(a_i))+v(a_i)).$$

如果对各个子问题 $KS(A_n\backslash a_i,W-w(a_i))$不再进行完整的推演,而只是分别进行贪心简约,那么就得到了近似比为 2 的多项式算法.进一步地,如果贪心简约是从 PRG 的第 k 层开始进行,那么算法的近似比将为 $1+1/k$,即 0-1 背包问题具有多项式时间的近似框架[24].

NP 完全问题并不意味着该问题的所有实例都是难以求解的.参数复杂性理论[25]就将问题的部分输入视为参数,那么在特定的参数范围内,问题也很可能是容易求解的.以最小顶点覆盖问题为例,给定图 $G=(V_m,E_n)$,要找出 V 中的一个最小顶点集,使得对于 E 中的每一条边 x,左端点 $x.a$ 和右端点 $x.b$ 至少有一个属于该顶点集,那么问题的规约为

$$MVC(V_m,E_n)\equiv(\mathbf{MIN}z:(\forall x:x\in E_n:x.a\in z\vee x.b\in z)\wedge z\in P(V_m):|z|) \tag{8}$$

令 $N(a)$表示与顶点 a 相邻的所有边集,那么约束条件可分解为

$$(e_n.a\in z\wedge(\forall x:x\in E_n\backslash N(e_n.a):x.a\in z\vee x.b\in z))\vee(e_n.b\in z\wedge(\forall x:x\in\backslash N(e_n.b):x.a\in z\vee x.b\in z)).$$

在此基础上,可对问题作如下推演:

$MVC(V_m,E_n)$

$\equiv(\mathbf{MIN}z:((e_n.a\in z\wedge(\forall x:x\in E_n\backslash N(e_n.a):x.a\in z\vee x.b\in z))\vee(e_n.b\in z\wedge(\forall x:x\in\backslash N(e_n.b):x.a\in z\vee x.b\in z)))\wedge z\in P(V_m)):|z|)$

≡[范围分裂]

$\min((\mathbf{MIN}z:e_n.a\in z\wedge(\forall x:x\in E_n\backslash N(e_n.a):x.a\in z\vee x.b\in z)\wedge z\in P(V_m):|z|),$

$(\mathbf{MIN}z:e_n.b\in z\wedge(\forall x:x\in\backslash N(e_n.b):x.a\in z\vee x.b\in z)\wedge z\in P(V_m):|z|))$

≡[拆分 V_m]

$\min((\mathbf{MIN}z':(\forall x:x\in E_n\backslash N(e_n.a):x.a\in z'\vee x.b\in z')\wedge z'\in P(V_m\backslash\{e_n.a\}):|z'\cup\{e_n.a\}|),$

$(\mathbf{MIN}z':(\forall x:x\in\backslash N(e_n.b):x.a\in z'\vee x.b\in z')\wedge z'\in P(V_m\backslash\{e_n.b\}):|z'\cup\{e_n.b\}|))$

≡[单点范围和范围分裂]

$\min((\mathbf{MIN}z':(\forall x:x\in E_n\backslash N(e_n.a):x.a\in z'\vee x.b\in z')\wedge z'\in P(V_m\backslash\{e_n.a\}):|z'|)+1,$

$(\mathbf{MIN}z':(\forall x:x\in\backslash N(e_n.b):x.a\in z'\vee x.b\in z')\wedge z'\in P(V_m\backslash\{e_n.b\}):|z'|)+1)$

≡[卷叠]

$\min(MVC(V_m\backslash\{e_n.a\},E_n\backslash N(e_n.a))+1,MVC(V_m\backslash\{e_n.b\},E_n\backslash N(e_n.b))+1)$

而对于判断图 $G=(V_m,E_n)$是否有度数小于 k 的最小顶点覆盖的参数化问题 $MVC(V_m,E_n,k)$,基于上述推演过程可得

$$MVC(V_m,E_n,k)\equiv\min(MVC(V_m\backslash\{e_n.a\},E_n\backslash N(e_n.a),k-1),MVC(V_m\backslash\{e_n.b\},E_n\backslash N(e_n.b),k-1)).$$

这样就得到了该问题的固定参数算法,由于问题简约至多只需要执行 k 层,得到的参数化算法的时间复杂度为 $O(2^k|V|)$.由于参数化算法可以方便地进行组合,对原始问题规约还可以再次基于生成函数的分解关系 $P(V_m)=P(V_{m-1})\cup\{z|z=z'\cup\{v_m\}\wedge z'\in P(V_{m-1})$进行如下推演:

$MVC(V_m,E_n)$

$\equiv(\mathbf{MIN}z:z:(\forall x:x\in E_n:x.a\in z\vee x.b\in z)\wedge(z\in P(V_{m-1})\vee(z=z'\cup\{v_m\}\wedge z'\in P(V_{m-1}))):|z|)$

≡[范围分裂]

$\min((\mathbf{MIN}z:(\forall x:x\in E_n:x.a\in z\vee x.b\in z)\wedge z\in P(V_{m-1}):v(z)),$

$(\mathbf{MIN}z':(\forall x:x\in E_n\backslash N(v_m):x.a\in z'\vee x.b\in z')\wedge z'\in P(V_{m-1}):|z'\cup\{v_m\}|))$

≡[卷叠和单点范围]

$\min(MVC(V_{m-1},E_n),(\mathbf{MIN}z':(\forall x:x\in E_n\backslash N(v_m):x.a\in z'\vee x.b\in z')\wedge z'\in P(V_{m-1}):|z'|+1))$

≡[卷叠]

$\min(MVC(V_{m-1},E_n),MVC(V_{m-1},E_n\backslash N(v_m))+1)$

将上述两步组合得到的算法效率为 $O(k|V|+2^kk^2)$;基于问题核心推演,还可以进一步提高算法效率至 $O(k|V|+1.32^kk^2)$乃至更高[26,27].

3 算法生成系统

我们设计了基于问题简约的组合优化算法生成系统,其主要由以下 5 个部件组成,结构如图 1 所示.

(1) 算法设计器:系统的核心部件,运用演算规则进行问题简约和算法生成;

(2) 知识库:有关问题基础结构的知识库,包括各种基本数据类型(如集合、序列、树、图等)的代数规约,其上的典型生成函数(如幂集函数、排列函数、子序列函数、子树/图函数等)及其分解关系等;

(3) 算法模式库:存放典型的问题简约模式及泛型算法;如果新问题的简约模式与库中某个模式同态,那么可以直接通过泛型实例化来得到其求解算法.表 2 中列出了系统预定义的主要抽象算法模式;

(4) 定理证明器:如果生成函数或约束函数的分解关系、贪心简约中的单调函数等是由用户自行指定的,那么定理证明器将对有关结果的正确性进行证明;

(5) 程序生成器:将抽象算法自动转换为可执行语言(如 C++,C#,Java 等)的算法程序,并针对特定目标平台进行代码优化.

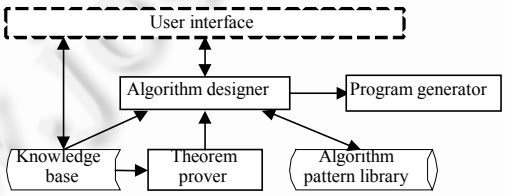


Fig.1 Basic structure of combinatorial optimization algorithm generation system

图 1 组合优化算法生成系统的基本结构

Table 2 Reduction-Based algorithm patterns and their application problems

表 2 简约算法模式及应用问题

Algorithm patte ıs	Application problems
Single singleton greedy reduction	Minimum spanning tree, compatible activity-selection, scheduling unit-time tasks with deadlines and penalties for a single processor
Multiple singleton greedy reduction	Single machine scheduling, Huffman coding tree, single-source shortest path, shuttle transportation
Sequencial reduction	Maximum sum, longest increasing subsequence, DNA sequence alignment, polygon decomposition
Binary reduction	0-1 knapsack, subset sum, maximum clique, minimum vertex covering, maximum-flow/minimum-cut
General reduction	Integer knapsack, traveling salesman, set covering, multiple vechile routing

4 结束语

算法问题是计算机科学中最为核心的问题之一.本文提出了一种基于代数规约的组合优化算法推演技术,它通过一组算法演算规则对问题进行简约,并基于简约关系构造正确高效的问题求解算法.该方法综合了动态规划、贪心、分支限界多种传统算法设计策略,并在传统方法与近似算法、参数化算法等 NP 难题的有效处理方法之间建立了联系,更加有利于人们对算法本质特征的理解.实验系统证明,该方法能够有效支持大量算法程序的自动生成,后续研究的重点将放在对并行算法自动生成的支持上.我们相信,随着最关键的算法设计形式化和自动化程度的不断提高,整个软件工程自动化的研究将有望取得新进展.

References:

[1] Xue JY. A unified approach for developing efficient algorithmic programs. Journal of Computer Science & Technology, 1997,12(4): 103−118.

[2] Helman P. An algebra for search problems and their solutions. In: Kanal L, Kumar V, eds. Proc. of the Search in Artificial Intelligence. Berlin: Springer-Verlag, 1988. 28−90.

[3] Helman P. A common schema for dynamic programming and branch and bound algorithms. Journal of the ACM, 1989,36(1): 97−128. [doi: 10.1145/58562.59304]

[4] Luan SM, Li W, Ma SH. Algorithm framework: An operational approach to algorithm relocation. Journal of Software, 1999,10(7): 679−684 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/10/679.htm

[5] Kumar V. A unified approach to problem solving search procedures [Ph.D. Thesis]. Department of Computer Science, University of Maryland, 1982.

[6] Kumar V, Kanal LN. A general branch and bound formulation for understanding and synthesizing and/or tree search. Artificial Intelligence, 1983,21(1-2):179−198. [doi: 10.1016/S0004-3702(83)80009-5]

[7] Kumar V. A general heuristic bottom-up procedure for searching and/or graphs. Information Science, 1991,56(1-3):39−57.

[8] Bird RS. An introduction to the theory of lists. NATO ASI Series F, 1987,36:5−42.

[9] Meertens L. Algorithmics towards programming as a mathematical activity. In: Bakker JW, Vliet JC, eds. Proc. of the CWI Symp. on Mathematics and Computer Science. 1986. 289−334.

[10] Backhouse RC. An exploration of the bird-meertens formalism. Technical Report, CS 8810, Department of Computing Science, Groningen University, 1988.

[11] Burstall RM, Darlington J. A transformation system for developing recursive programs. Journal of the ACM, 1977,24(1):44−67. [doi: 10.1145/321992.321996]

[12] Bibel W, Hornig KM. LOPS—A system based on a strategical approach to program synthesis. In: Biermann AW, Guiho G, Kodratoff Y, eds. Proc. of the Automatic Program Construction Techniques. New York: MacMillan, 1984. 69−89.

[13] Neugebauer G, Fronhöfer, B, Kreitz C. XPRTS—An implementation tool for program synthesis. In: Metzing D, ed. Proc. of the 13th German Workshop on Artificial Intelligence, Vol.216. Informatik-Fachberichte, 1989. 348−357.

[14] Xu JF, Dai M, Lü J. Algorithm design automation system NDADAS. Journal of Computer Research and Development, 1990,27(2):1−5 (in Chinese with English abstract).

[15] Lü J. Framework of algorithm correctness in NDADAS. Science in China, Series F, 1991,34(7):875−884.

[16] Zhang JZ, Fei ZM. Automatic generation of software based on scheme. Journal of Computer Research and Development, 1992,29(6):1−6 (in Chinese with English abstract).

[17] Smith DR. KIDS: A knowledge-based software development system. In: Lowry M, McCartney R, eds. Proc. of the Automating Software Design. Cambridge: The MIT Press, 1991. 483−514.

[18] Smith DR. Designware: Software development by refinement, high integrity software. IEEE Computer, 2001,20(4):10−19.

[19] Xue JY. PAR method and its supporting platform. In: Proc. of the 1st Asian Working Conf. on Verified Software (AWCVS 2006). 2006. 29−31.

[20] Zheng YJ, Xue JY. A problem reduction based approach to discrete optimization algorithm design. Computing, 2010,88(1-2): 31−54. [doi: 10.1007/s00607-010-0085-0]

[21] Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to Algorithm. 2nd ed., New York: McGraw-Hill, 2001. 339−341.

[22] Bird RS, de Moor O. From dynamic programming to greedy algorithms. In: Proc. of the IFIP TC2/WG 2.1 State-of-the-Art Report on Formal Program Development. LNCS 755, 1993. 43−61.

[23] Edmonds J. Matroids and the greedy algorithm. Mathematical Programming, 1971,1(1):171−236. [doi: 10.1007/BF01584082]

[24] Sahni S. Approximate algorithms for the 0/1 knapsack problem. Journal of the ACM, 1975,22(1):115−124. [doi: 10.1145/321864.321873]

[25] Downey RG, Fellows MR. Fixed-Parameter tractability and completeness I: Basic theory. SIAM Journal of Computing, 1995,24(4): 873−921. [doi: 10.1137/S0097539792228228]

[26] Chen JE, Huang ZX, Kanjd IA, Xia G. Polynomial time approximation schemes and parameterized complexity. Discrete Applied Mathematics, 2007,15(2):180−193. [doi: 10.1016/j.dam.2006.04.040]

[27] Li SH, Wang JX, Chen JE. Kernelization techniques and its applications to parameterized computation. Journal of Software, 2009, 20(9):2307−2319 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3593.htm [doi: 10.3724/SP.J.1001.2009.03593]

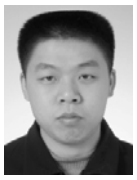
附中文参考文献:

[4] 栾尚敏,李未,马绍汉.算法框架:算法重定位的一种可操作的方法.软件学报,1999,10(7):679−684. http://www.jos.org.cn/1000-9825/10/679.htm

[14] 徐家福,戴敏,吕建.算法自动化系统 NDADAS.计算机研究与发展,1990,27(2):1−5.

[16] 张家重,费宗铭.基于算法构架的软件自动产生.计算机研究与发展,1992,29(6):1−6.

[27] 李绍华,王建新,陈建二.参数计算中核心化技术及其应用.软件学报,2009,20(9):2307−2319. http://www.jos.org.cn/1000-9825/3593.htm [doi: 10.3724/SP.J.1001.2009.03593]



郑宇军(1979－),男,福建莆田人,博士,高级工程师,主要研究领域为运筹学,自动化软件工程.

凌海风(1975－),女,博士,副教授,主要研究领域为软件工程方法学.

薛锦云(1947－),男,教授,博士生导师,主要研究领域为软件形式化和自动化.