求解组合优化问题的一种方法—分枝定界法

汪祖柱[1,2], 程家兴[1]
(1.安徽大学 计算智能与信号处理教育部重点实验室,安徽 合肥 230039;
2.安徽大学 管理学院,安徽 合肥 230039)

摘 要:较为详细地分析了分枝定界法的算法特征和过程,讨论了以该算法求解具体优化问题时所要采取的算法策略。笔者结合简单、具体的例子说明了上述过程,并且也说明了在实际应用该算法时,根据问题的局部信息和其它启发算法求解问题的必要性。

关键词:组合优化;分枝定界法;算法
中图分类号:TP301.6 **文献标识码**:A **文章编号**:1000-2162(2004)01-0010-05

组合优化问题是相对基于连续变量求解的函数优化问题来进行讨论的[1]。所谓组合优化是指在离散的、有限的数学结构上,寻找一个满足给定约束条件并使其目标函数值达到最大或最小的解。从一般意义上说,许多最优化问题的研究都属于组合优化,比如研究和应用较多的规划问题,从求解问题的诸多方法考虑,也属于组合优化的领域。如今,求解组合优化问题遍及许多学科领域,如计算机科学、应用数学、管理科学等其他相关学科。而且,随着求解问题的算法研究和实现算法的技术提高,一些新的组合优化问题不断出现,求解方法也得到快速发展[6]。本文就求解组合优化问题的基本方法——分枝定界算法(branch-and-bound algorithm),来探讨该算法的分析及应用策略。

1 算法概述

分枝定界算法(Branch-and-Bound Algorithm,简称 B&B)由 Land Doig 等人于本世纪六十年代提出[2]。其算法思想不仅适用于表达成整数线性规划(或者混合整数线性规划)的问题,也适用于几乎任何组合最优化问题[1]。它采用了类似分而治之的算法策略,在分析一个组合最优化问题的一切可行解的过程中,采取了必要的限制条件,设法排除可行域中大量非最优解区域,从而能够有效求解一些规模较大的问题。

1.1 算法特征

设一个最优化问题的一个实例为(F,f),其中 F 为一有限集合或可行域,f 为费用函数或映射,即求解下列问题

$$\text{minimize } f(x),\ \text{subject to } x \in F \tag{1}$$

在以 B&B 方法来求解(1)时,原问题将被分解为一个或多个子问题,其中与子问题对应的可行域为 $F_1,\cdots,F_k$,$F_i \subseteq F(i=1,\cdots,k)$。第 i 个子问题是对 $x \in F_i$,求解 minimize $f(x)$。接

收稿日期:2003-06-23
作者简介:汪祖柱(1969-),男,安徽肥西人,安徽大学讲师,在职博士研究生.

着按同样的方式将可行域为 F_i 的子问题再分解为若干个子问题，直到某个子问题容易求解（求到原问题的一个隐含最优解或不含最优解）为止。整个求解过程可以用下式表示：

$$\min_{x\in F} f(x) = \min_{1\leqslant i\leqslant k} \{\min_{x\in F_i} f(x)\} \tag{2}$$

上述过程中，对任意一个可行域为 $F' \subseteq F$ 的子问题，即对 $x \in F'$ 求 minimize $f(x)$，设其存在对应的一个下界为 $L(Y)$，使得 $L(Y) \leqslant \min_{x\in F'} f(x)$（$Y$ 见后面叙述）。

1.1.1　算法步骤　算法过程中，将所有待分解或探查的活点（子问题）存储于集合 activeset 中，U 中存放目前发现的最优解值。

（1）初始化（initializing step）。U = ∞。尽可能去掉一些明显的非最优点，将其余的可行解集作为一个子集合．转到 step 2。

（2）分枝（branching step）。采用某种分枝规则，从目前的若干可行解子集中选择一个子集，将其分解为若干子集合。

（3）定界（bounding step）。对每个新子集 Y，计算 $L(Y)$。

（4）探查（fathoming step）。根据 $L(Y)$ 的估计值进行判断，决定是否进一步分解子集 Y，情况如下：(a) $L(Y) \geqslant U$；(b) Y 中不含可行解，从可行域中排除 Y；(c) 对问题 $\min_{x\in Y} f(X)$ 求到最优解 x'，那么 $L(Y) = f(x')$。如果(a)成立，那么从可行域中去掉集合 Y；如果(a)不成立，则令 $U = L(Y)$，将 x′作为当前最好解，然后，对其它活点（子问题）再根据(a)进行如上判断。

（5）停止规则（stopping step）。如果没有活点（子问题）待处理，即 actieset 中不含点集，算法终止，当前所获得的最好解为最优解。否则，返回到 step 1。

一般意义上的 B&B 算法过程如下所示，该算法描述表示一个基本分枝定界算法。任何时刻集合 activeset 存储各个活点，变量 U 存储任何给定时间得到的最好解（U 是最优值的一个上界）。

```
begin
activeset: = {0}(comment:"0"表示原问题);
U: = ∞;
currentbest: = anything;
while activeset 非空 do
begin
    选择一个分枝点出 k∈activeset;
    从 activeset 中去掉点 k;
    生成点 k 的各个儿子 j, j = 1,2,…,n_k 及其对应的下界 z_j;
    forj = 1,…,n_k,do
      begin
      if zj≥U,then 杀死儿子 j
      else if 儿子 j 是某种符合要求的解(如对于整数规划问题,其解的形式是儿子 j 是一全整数解)。
      then
```

U: = zj, currentbest: = 儿子 j;
else 把儿子加到 activeset 中
end
end
end

1.1.2 例子:一个背包问题 一个徒步旅行者旅行时准备带一些物品,物品的种类、价值和重量分别如下表所示。问题是,他应当选择哪些物品,使得所选物品的总价值至少为9,而总的重量最小。问题的数学表示为:

$$\min \sum_{i=1}^{4} x_i w_i, \quad s.t. \sum_{i=1}^{4} x_i w_i \geqslant 9, x_i \in \{0,1\}, i = 1,2,3,4$$

及(i =1,2,3,4)的有关数据见表1。

表1 背包问题的有关数据

i	1	2	3	4
v_i	5	5	4	2
w_i	5	6	3	1
w_i/v_i	1.0	1.2	0.75	0.5

集合{1,2,3,4}的16个子集中的每一个都可能为问题的解.然而,只有8个是可行解.下面来考虑这个问题(参考图1所示的搜索树)。

(1) 开始时,X 作为唯一的一个子集,按照下标序选择物品直到物品总价值不小于9为止,从而得到 $L(X) = 11$;

(2) 将 X 按照旅行者是否选择物品1而分解为 X_1(物品1被选择)和 X_0(没有选择物品1)两个子问题.很显然 $L(X_0) = 1, L(X_1) = 5$,所以 $activeset = \{X_0, X_1\}$;

(3) 将 X_0 进一步分解为 X_{01} 和 X_{00},X_{01},X_{00} 分别表示物品1没被选择的前提下,按物品2是否被选择进行分解(下面的每一步分解与此意思相同,不再解释).X_{00} 不含可行解,被排除.$L(X_{01}) = 6$, $activeset = \{X_{01}, X_1\}$;

(4) 将 X_1 分解为 X_{11} 和 X_{10},计算得 $L(X_{10}) = 5, L(X_{11}) = 11$.根据($a$),$X_{11}$ 被排除,$activeset = \{X_{01}, X_{10}\}$;

(5) 将 X_{10} 分解为 X_{101} 和 X_{100},根据(b),X_{100} 被排除.解 X_{101} 得 $x' = \{1,3\}, f(x') = 8$.按照(c),$U = 8$, $activeset = \{X_{01}\}$;

(6) 将 X_{01} 分解为 X_{011} 和 X_{010},由 $L(X_{011}) > U$,按照(a),X_{011} 被排除,至于 X_{010} 不包含可行解,根据(b)可以排除;

(7) 至此,$activeset$ 为空,运算结束.得到最优解 $x' = \{1,3\}$。

注1 该例的算法过程中,在运算所作的迭代次数与计算界估计的运算量之间作了平衡处理,比如,在计算各个界L时,本来可以根据物品的具体信息使用贪婪算法(greedy algorithm)来计算,即在计算 $L(X_0)$ 时,根据物品2,3和4的 w_i/v_i 值(分别为6/5,3/4,1/2),所以不选择物品1时,使得背包所装物品价值不小于9且物品总重量最轻,那么首先选择物品4,其

次 3,而物品 2 只需 3 个价值单位即可。这样 $L(X_0) = 1 + 3 + 3(6/5) = 7.6$,以此类推,$L(X_1) = 5 + 1 + 2(3/4) = 7.5$,$L(X_{01}) = 6 + 1 + 2(3/4) = 8.5$,所以在 step 5 后,$X_{01}$ 便能被排除,从而 step 6 将是不必要的。按这样的方法进行计算可减少迭代次数,然而界估计的运算量增加了。

注 2　若在求解时,需要所有的最优解,那么可以以 $L(Y) > U$ 替代(a)。这样,若 $L(Y) = U$时,将对应的解保存在解集中,若 $L(Y) < U$,则将当前解集中所有解以新的最好解代之即可。

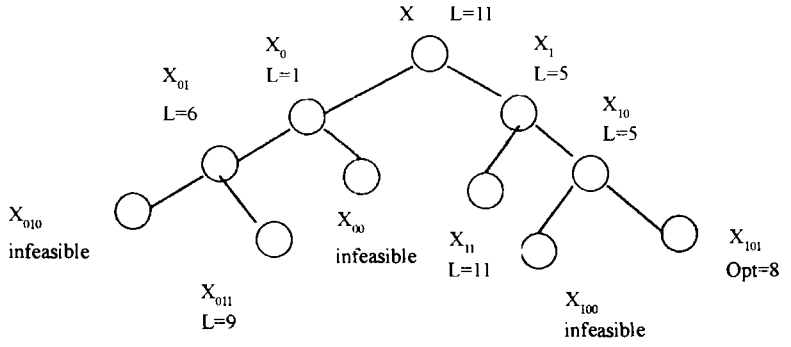

图 1　求解背包问题的搜索树

1.2　算法策略分析

(1) 分枝(branching step)。一般情况下,常用的两个分枝规则是:

最低下界优先(best bound rule)。即选择带有最低下界(lowest bound)的子集进行分枝,采用这种方法的目的是希望它能提供寻求最优值的最好机会,并且通过探测来排除大量非最优点。不足之处是对于一个规模较大的问题来说,每次寻找符合要求的子集需要花费相当的时间。另外,如果算法过程中存储不是个决定因素,那么该方法也是比较合理的。

最新下界优先(newest bound rule)。这种方法是选择最新形成的子集来进行分枝,类似于深度优先搜索(dfs)。该方法的优点在于避免了算法流程反复跳转于对应的搜索树中,并且节省了算法在定界时的消耗,从而可以提高算法的效率;缺点在于,如果当前考察的子问题对应的可行解与最优解差别很大时,那么将会产生大量非必要的界估计运算。

(2) 定界(bounding step)。在处理具体的优化问题时,下界的计算需采用具体的方法和一些启发式算法来进行估计,如在处理整数规划的一类问题时,常采用的方法有 Lagrangian 松弛法和线性规划松弛法[4]。

运用上述策略解决问题时,往往要根据问题的自身特点和求解的要求来决定采取何种方法,有时是若干方法的结合使用。至于在以 B&B 方法来求解问题时,如何运用适当的方法来具体的优化问题,一个通常的问题在于方法是综合考虑了算法的时间和空间及求解质量的要求。

2　结束语

本文对分枝定界法作了具体分析,分枝定界法是一种求解组合优化问题的有效方法。在实际应用中,将算法的特征及其过程与求解的具体问题结合考虑是十分重要的。特别在

处理一个具体复杂的问题时,运用该算法时,有时要结合其它算法才能有效求解问题,如文献[5]便是在结合了人工智能的搜索算法思想,来提高 B&B 方法的求解效率。

参考文献:

[1] C H Papadimitriou, K Steiglitz. Combinatorial Optimization: Algorithms and Complexity[M]. Englewood Cliffs, Newjersey: Prentice Hall, 1982. 559 - 566.

[2] A kaufmann. Integer and Mixed Programming: Theory and Applications[M]. Academic Press, Inc(London), 1982. 95 - 106.

[3] 刘明,吴唤群. 资源有限——工期最短的分枝定界算法[J]. 系统工程,1999,117(3):72 - 75.

[4] 刑文训,谢金星. 现代优化计算方法[M]. 北京:清华大学出版社,1999. 248 - 282.

[5] 杜江,孟香惠,等. 一种改进的分枝定界算法[J]. 数学杂志,1998,18:55 - 58.

[6] 吴小培,贯勤云. 一种提高 BP 算法学习速度的有效途径[J]. 安徽大学学报(自然科学版),1998, 22(3):64 - 67.

An algorithm on solving combinatorial optimization problems —branch - and - bound method

WANG Zu - zhu[1,2], CHENG Jia - xing[1]

(1. Ministry of Educational Key Laboratory of Intelligent Computing & Signal Processing, Anhui University, Hefei 230039, China; 2. Management School, Anhui University, Hefei 230039, China)

Abstract: In this paper, some characteristics and processing steps about branch - and - bound method(B&B) are analyzed in more details, and some strategies, used to solving optimal problems by B&B, are discussed. One example is supplied to explain the method in practical application, and it is necessary to combine the method with some heuristics and local information from the discussed problem to solve the problem.

Key words: combinatorial optimization; branch - and - bound method; algorithm