双指针线性搜索算法的控制策略与编程

何伯方

(浙江技术师范专科学校电气工程系)

摘 要 给出了双指针线性搜索算法与4种数学模型的关系及关于该算法原理的数学证明。提出指针徘徊控制策略以确保最优解不丢失,从而使该算法推广到含重复表元的有序表,还给出相应的程序框架,其中采用动态邻域来加快抛弃无关数据以提高计算效率。

关键词 数学模型 重复表元 指针徘徊 动态邻域

0 引 言

文献[1]为离散规划[2-4]提出一种双指针线性搜索算法并定义了建模概念: 函数分解代换,还指出4种数学模型适合采用该算法但未进一步解释。该算法的意义在于: 局部函数f_A和f_B可由函数分解得到或另行构造,只要保证最优解不丢失。于是,只须按照双指针性搜索算法来计算 $f_A \bigstar f_B$,不必计算相对复杂的函数f并有较高的计算效率。

通过深入研究,发现该算法对不含重复表元的有序表有效,但在处理含重复表元的有序时却不太稳定,有时会漏掉部分最优解;而在实际工程问题中又确实会遇到含重复表元的有序表。针对这一情况,本文提出指针徘徊控制策略,能保证不丢失任何最优解,从而把该算法推广到含重复表元的有序表,并提出动态邻域技术来加快抛弃无关数据以提高计算效率。

1 双指针线性搜索算法的4种数学模型与证明

为方便后面的讨论,下面给出该算法与4种数学模型的关系及严格的数学证明,并作为对文献[1]的补充。

1.1 4种数学模型

文献[1]把二元★运算定义为普通四则运算, 相应地, 函数分解代换 $f = f_A \bigstar f_B$ 就有$f = f_A + f_B$、$f = f_A - f_B$、$f = f_A \cdot f_B$和$f = f_A \div f_B$四种类型,分别称为函数f的(A + B)型、(A − B)型、(A · B)型和(A ÷ B)型函数分解代换。研究证明,这4种数学模型均适合采用双指针性搜索算法。表1给出函数分解代换的4种模型与双指针线性搜索算法的对应关系,

收稿日期: 1997年1月23日

约定其中 (A · B)型和 (A ÷ B)型局部解集有序表的表元皆大于 0。

表 1 双指针线性搜索算法与四种模型的关系

函数分解代换类型	局部解集有序表	指针初始位置	指针移动方向
$f = f_A \div f_B$	T_A 和 T_B均	T_A和T_B的指	$f < f_S$
$f = f_A - f_B$	为升序表	针I 和J 均	正向移动I
$f = f_A \cdot f_B$	T_A 为升序表	为头指针	$f \geqslant f_S$
$f = f_A + f_B$	T_B 降 序 表		正向移动J

1.2 双指针线性搜索算法的原理

顾名思义,这种算法使用两个指针对有序表进行线性搜索。

设:局部函数 f_A和 f_B 的非空局部解集分别为$\{f_A\}$和$\{f_B\}$,以 (A + B)型函数分解代换为例,双指针线性搜索算法的离散规划数学模型为:

极小化目标函数 $(f - f_S)^2$ (1)

满足约束条件 $f = f_A + f_B,\ f_A \in T_A,\ f_B \in T_B$ (2)

$f_S > 0$ (3)

其中:f_S是 f的标准值,T_A和 T_B 分别为$\{f_A\}$和$\{f_B\}$构成的有序表。

为方便叙述,所有表项统一用 ① ~ ⑨ 来表示并设 $T_A = \{①, ②, ③, ④\}$, $T_B = \{⑨, ⑧, ⑦, ⑥, ⑤\}$。其中,① ~ ④ 为被加数升序列,⑨ ~ ⑤ 为加数降序列。

又设:I 为被加数指针,J 为加数指针。

恰当地为 f_S选择上限 f_{Smax}和下限 f_{Smin},即设置取样窗口。通常,$f \neq f_S$。

将指针 I 、J 置于各自的表头,I 指向①,J 指向⑨ 。注意被加数的值偏小或加数的值偏大是正向移动各自指针的依据。一般地,若 $f < f_S$,移动指针 I;若 $f \geq f_S$,移动指针 J。

求解 f的计算过程如下(参见图 1):

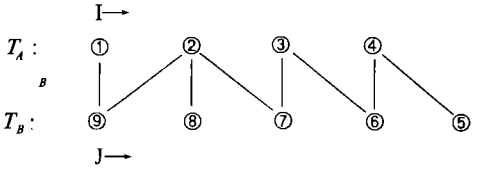

图1 表示双指针线性搜索的一种可能的情况,其中| / 和\ 各表示一次"取数—计算—筛选",I→J→表示两个指针及移动方向

首先计算 f = ① + ⑨。若 $f \leq f_{Smin}$,说明被加数偏小,应正向移动被加数指针使被加数增大,于是 I 指向②,即抛弃①。接着计算f = ② + ⑨。若$f \geq f_{Smax}$,说明加数偏大,应正向移动加数指针使加数减小,于是 J 指向⑧,即抛弃⑨ ……

对落入窗口的f进一步筛选便得到最优解。

下面证明抛弃数据的正确性、最优解不会丢失和该算法的时间复杂度为 $O(n)$。

证明 i) 根据上面的假定,

因为 ① + ⑨ $\leq f_{Smin}$且 ⑤ < ⑥ < ⑦ < ⑧ < ⑨,

必有 ① + ⑤ < ① + ⑥ < ① + ⑦ < ① + ⑧ < ① + ⑨ $\leq f_{Smin}$,

所以 ① 可不必再与 ⑤、⑥、⑦ 或 ⑧ 进行计算,即抛弃 ① 是正确的;

因为 ② + ⑨ $\geq f_{Smax}$且 ② < ③ < ④,

必有 ④ + ⑨ > ③ + ⑨ > ② + ⑨ $\geq f_{Smax}$,

所以 ⑨ 可不必再与 ③ 或 ④ 进行计算,即抛弃 ⑨ 也是正确的;

同理 抛弃其它数据也是正确的。

因此 被抛弃的数据与最优解无关。

ii) 设 f_S邻域的选取是恰当的,必有 f 满足 $f_{Smin} < f < f_{Smax}$,

因此 最优解不会丢失。

iii) 设有序表 T_A和 T_B分别含 m 项和 n 项表元,

因为 指针在表内的每次移动,会使另一表的某项多参与一次运算且指针 I 和 J 分别在各自的表内最多移动 $(m-1)$次和 $(n-1)$次,

所以 T_A的 m 项表元最多增加 $(n-1)$次运算,T_B的 n 项表元最多增加 $(m-1)$次运算。

因此 采用双指针线性搜索算法求 f 最多只需计算 $(m+n-1)$次;$m = n$ 时,最多只需计算 $(2n-1)$次。双指针线性搜索算法的时间复杂度为 $O(n)$。

2 有序表含重复表元的控制策略——指针徘徊

当有序表含重复表元时,指针的移动规则可能使部分最优解丢失。

图 2 表示有序表含重复表元的三种情况。

在图 2(a)中,当第一个 X 与 Y 计算后,若 X★Y $\geq f_{Smax}$(或 X★Y $\geq f_S$)则移动指针 J,使其余 X 未进行 X★Y 计算。在图 2(b)中,当 X 与第一个 Y 计算后,若 X★Y $\leq f_{Smin}$(或 X★Y $\leq f_S$)则移动指针 I,使其余 Y 未进行 X★Y 计算。在图 2(c)中,移动指针一定会使部分

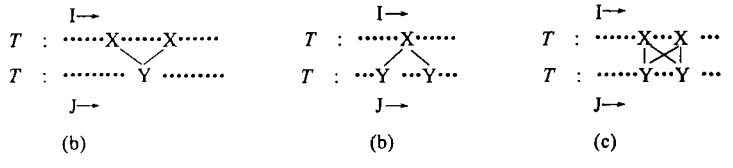

图2 (a)、(b)和(c)表示有序表中含重复表元的三种情况,其X…X和Y…Y表示若干重复表示,表元间的连续表示应该进行运算

X 或者部分 Y 不参与 X★Y 计算。当图 2 中的 X★Y 不是当前最优解时,漏掉部分重复表元 X 或 Y 是无关紧要的。若 X★Y 恰好是当前最优解,则不允许漏掉任一 X 或 Y。

定义 1 设 X…X 和 Y…Y 为重复表元且图 2(a)中的 X…X 和 Y、(b)中的 X 和 Y…Y 及 (c) 中的 X…X 和 Y…Y 均与当前最优解有关.称遍历这些数据的指针操作为指针徘徊,并称这样的数据区为指针徘徊区。

指针徘徊区内数据的特殊性:计算第一对数据 X★Y 且发现是当前最优解时,其余 X★Y 必为当前最优解,不需再计算。通过遍历指针徘徊区,即可直接获得区内的所有当前最

优解。控制策略如下:一旦发现当前最优解,

1) 确定有序表 T_A 当前重复表元数;

2) 确定有序表 T_B 当前重复表元数;

3) 采用二重循环实现指针徘徊,将指针徘徊区内所有当前最优解送入队列。

经此特殊处理后,不仅确保最优解不丢失,而且指针徘徊区之外的其它表元仍可按照无重复表元的有序表来处理,因而使双指针线性搜索算法推广到含重复表元的有序表。

3 双指针线性搜索算法的程序算法

本程序算法符合结构化设计原理,采用编译型 BASIC 语言的格式,并在括号中给出必要的注释。根据表 1,有序表 T_A 都为升序表,对于 (A − B)型和 (A ÷ B)型,要求有序表 T_B 为升序表,而对于 (A + B)型和 (A · B)型,要求有序表 T_B 为降序表。

若邻域选得过小,一轮计算可能得不到最优解,必须放宽邻域重新计算。采用"动态邻域"能避免发生这种情况,而且不必精心选择邻域。所谓"动态邻域",即邻域的初值取得大一些(以免丢失最优解),以后根据新当前最优解的偏差来缩小邻域和取样窗口。动态邻域技术能加快抛弃无关数据,提高计算效率。实践证明,取样窗口应略大于邻域。

模块 1:读入有序表;(有序表的磁盘文件由另一程序生成。)

模块 2:准备标准值 Fs;

模块 3:寻找最优解

```
1.  L = 0    (初始化最优解队列指针)
2.  F0 = 0.005 * Fs  (初始化邻域)
3.  Fsmin = Fs − 1.1 * F0
4.  Fsmax = Fs + 1.1 * F0
5.  I=0    (初始化 TA 的指针)
6.  J = 0  (初始化 TB 的指针)
7.  WHILE I <= Imax AND J <= Jmax
7-1.  F = TA(I)★ TB(J)
7-2.  IF F < Fsmax AND F > Fsmin THEN
  A.     F1 = F − Fs          (计算偏差)
  B.     F2 = ABS(F1)
  C.     IF F2 < F0 THEN (新解更优)
  C-1.     L = 0  (抛弃原当前最优解)
  C-2.     F0 = F2          (修改邻域)
  C-3.     Fsmin = Fs − 1.1 * F0
  C-4.     Fsmax = Fs + 1.1 * F0
           END IF
  D.     IF F2 = F0 THEN(是当前最优解)
  D-1.     II = I:JJ = J  (确定指针徘徊区)
```

```
D-2.    WHILE TA(I) = TA(II + 1)
          II = II + 1
        WEND
D-3.    WHILE TB(J) = TB(JJ + 1)
          JJ = JJ + 1
        WEND
D-4.    FOR IP = I TO II  (指针徘徊)
          FOR JP = J TO JJ
D-4-1.      L = L + 1
D-4-2.      新最优解进队列
          NEXT JP
        NEXT IP
D-5.    I = II:J = JJ
      END IF
    END IF
7-3.  IF F >= Fs (通常 F ≠ Fs)
      J = J + 1(是正偏差,移动指针 J)
    ELSE
      I = I + 1(是负偏差,移动指针 I)
    END IF
  WEND
8.  显示或打印队列中的所有最优解;
9.  END
```

4 问题讨论

1. 以上的讨论曾约定 (A · B) 型和 (A ÷ B)型函数分解代换的所有表元都大于 0。若表元为其它情况,需对有序表的表元分段处理。

2. 设置一个队列是必要的。研究中曾发现有多至 9 个最优解的情况。

5 结束语

在解决工程上的实际离散规划问题时常碰到最优解不是唯一的。选用一个方案后,其余的最优解就是备用方案。当某个齿轮或部件磨损或损坏时,就需要采用备用方案。因此,指针徘徊控制策略具有明确的实际意义和价值并使双指针线性搜索算法推广到含重复表元的有序表。指针徘徊与动态邻域技术也使该算法更加完善和实用。本文提供的程序算法简明、实用,便于推广和移植,也可植入到自动化机床中。

参考文献

1 何伯方.双指针线性搜索算法.软件,1996,(6):5 — 8

2 陈宝林.最优化理论与算法.北京:清华大学出版社,1989,8

3 席少霖,赵凤治.最优化计算方法.上海:上海科学技术出版社,1983,8

4 吴文江,袁仪方.实用数学规划.北京:机械工业出版社,1993,3

5 上海第一机床厂 Y3150 滚齿机使用手册.1986

CONTROL STRATEGY FOR LSADP AND PROGRAMMING

He Bofang

(Department of Electric Engineering, Zhejiang Teachers' College of Technology)

ABSTRACT The relationship between four types of mathematical model and the Linear Search Algorithm by Dual-Pointer (LSADP) was revealed, the principle of LSADP was justified, the control strategy of pointers-pacing up and down was provided to provent from losing any optical solution, and the frame of program was demonstrated, in which the technique of dynamic neibourhood was adopted to more rapidly exclude the don't-care data for advancing computational efficiency.

KEYWORDS mathematical model duplicate table element pointers-pacing up and down dynamic neibourhood