DESENVOLVIMENTO DE ALGORITMOS APROXIMATIVOS POR ACERCAMENTO: ESPECIFICAÇÃO FORMAL

Laira V. Toscani

Departamento de Informática - UFRGS
Atualmente na Universidade Nova de Lisboa
Fac. de Ciência e Tecnologia - Depto. de Informática
Grupo de Prog. em Lógica e Int. Artificial
2825 - Monte da Caparica - Portugal

Paulo A. S. Veloso

Departamento de Informática - PUC/RJ
R. Marquês de São Vicente 225
22453 - Rio de Janeiro - Brasil

RESUMO

Acercamento é um método de desenvolvimento de algoritmos aproximativos para uma classe de problemas de maximização, que satisfaz um dado requerimento de exatidão. A especificação formal apresentada utiliza tipos abstratos de dados.

ABSTRACT

Rounding is a method for the development of approximative algorithms to a class of maximization problems that satisfies a given accuracy requirement. The method is formally specified through abstract data types.

1. INTRODUÇÃO

O Acercamento é um método de desenvolvimento de algoritmos aproximativos (mdaa) para um certo problema de otimização que será definido na secção seguinte. Horowitz([HOR 78]) chama este mdaa de "Rounding". É um método interessante porque dado um requerimento de exatidão e regras de dominância, gera um algoritmo aproximativo polinomial no tamanho da entrada e inversamente polinomial ao requerimento de exatidão. Isto é, o algoritmo resultante é polinomial e gera soluções "satisfatoriamente próximas" da solução ótima.

A especificação formal do método consiste de um programa abstrato e um conjunto de axiomas. O programa abstrato mostra a estratégia do método. Os axiomas definem o inter-relacionamento das funções e dos predicados e permitem a verificação da correção do programa abstrato. A abstração é desenvolvida em dois níveis.

Para aplicar esta estratégia a uma instância do problema é suficiente escrever um módulo de implementação definindo as funções abstratas em termos do problema. Com isto, ter-se-á um algoritmo aproximativo, que é uma instância do programa abstrato, que satisfaz o requerimento de exatidão (ξ), cuja complexidade é polinomial em $\frac{n}{\xi}$, em que n é o tamanho da instância.

2.DEFINIÇÃO DO PROBLEMA

Calcular $\max_{x} \sum_{i=1}^{n} p_i x_i$ (função objetivo) restrito a $\sum_{i=1}^{n} a_{ij} x_i \leq b_j$, $a_{ij} \geq 0$ e $1 \leq j \leq n$; $x_i = 0$ ou 1, $p_i \geq 0$ e $1 \leq i \leq n$.

Definições:

Seja $I=(P,A,B)$, onde $A=(a_{ij})$ é uma matriz nxn, $B=(b_1,b_2,\ldots,b_n)$, $P=(p_1,p_2,\ldots,p_n)$ e $a_{ij}, b_i, p_j \geq 0$, $1 \leq i,j \leq n$. Seja $s=(s_1,s_2,\ldots,s_i)$, onde $s_j=0$ ou 1, $1 \leq j \leq i \leq n$.

- I é uma instância do problema.
- n é o tamanho de I.
- s é uma solução parcial possível de I. Se i=n, então s é uma solução possível e se além disso $\sum_{j=1}^{i} a_{jk} s_j \leq b_k$ $1 \leq k \leq i$, então s é uma solução parcial viável de I.
- $FO(I,s,i)=\sum_{j=1}^{i} p_j s_j$. Se s é a solução ótima de I restrito a i variáveis, $FP^*(I,i)=\sum_{j=1}^{i} p_j s_j$ e $FP^*(I,n)=F^*(I)=\sum_{j=1}^{n} p_j s_j$, em que s é a solução ótima para I.
- $S^{(k)}$ com $1 \leq k \leq n$ é o conjunto das soluções viáveis de I restrito às k primeiras variáveis.

3.DEFINIÇÃO DO MÉTODO DE ACERCAMENTO

O método consiste em transformar a instância original I em uma instância I', cuja solução está próxima da solução ótima da instância original, transformar I' em uma instância I", que acelera o processo de construção de $S^{(0)}$, $S^{(1)}$,..., $S^{(n)}$, sem alterar a solução ótima. Em $S^{(n)}$ é calculada a solução ótima de I" (e de I') e transportada para I.

Características do Método

Seja I uma instância dada de tamanho n, I' e I" as instâncias calculadas pelo método, ξ o requerimento de exatidão, s a solução de I" encontrada através do método e s^* a solução ótima de I. Então:

- a solução encontrada está satisfatóriamente próxima da solução ótima, isto é, $|(F^*(I)-F^*(I"))/F^*(I)| \leq \xi$;

- a solução encontrada s é a solução ótima das instâncias I' e I";

- os métodos de transformação de I em I' e de I' em I" tem complexidade linear no comprimento de I.

- a cardinalidade do conjunto de soluções parciais $S^{(k)}$ é da ordem $k(n/\xi)$, com $k=1,2,\ldots,n$. O que vai garantir uma complexidade total do método polinomial em n/ξ.

4.ESPECIFICAÇÃO FORMAL DO MÉTODO

A especificação formal é constituida do programa abstrato e do conjunto de axiomas.

Definição dos Domínios:

PO - domínio de instâncias do problema. Se $I \in PO$ então $I = (P,A,B)$, onde $A=(a_{ij})$ é uma matriz nxn , $B=(b_1,b_2,\ldots,b_n)$ e $P=(p_1,p_2,\ldots,p_n)$.

$\Re$ - domínio das entradas adicionais ξ e Lb; ξ é o requerimento de exatidão; Lb estimativa inferior para o valor ótimo da função objetivo.

RD - domínio das regras de dominância de soluções viáveis.

SP - domínio das soluções parciais: conjunto de conjuntos de i-uplas ($0 \leq i \leq n$), soluções possíveis do problema restrito às i primeiras variáveis.

SL - domínio das soluções: conjunto das n-uplas, soluções possíveis para o problema.

$\aleph$ - contradomínio da função tamanho, que define o tamanho da instância do problema.

V - domínio dos valores possíveis da função objetivo.

Funções Auxiliares:

- tamanho : PO $\rightarrow \aleph$, calcula o tamanho da instância do problema considerado.

- transforma1 : PO x $\Re^2$ $\rightarrow$ PO , transforma uma instância I, considerando Lb e ξ, em uma nova instância I' cuja solução está próxima da solução da instância original I.

- transforma2 : PO x $\Re^2$ $\rightarrow$ PO , transforma a instância I' em uma nova instância I" com mesma solução.

- inicializa : X^0 $\rightarrow$ SP , inicializa o conjunto de soluções viáveis para o caso de zero variáveis ($X^0=\{\varepsilon\}$, ε string vazio).

- calculaS : PO x SP x RD x $\aleph$ $\rightarrow$ SP , calcula o conjunto de soluções parciais viáveis, obtido considerando o conjunto de soluções parciais dadas, incluindo uma variável a mais no problema considerado e fazendo as eliminações permitidas pelas regras de dominância.

- calculaSO : PO x SP $\rightarrow$ SL, escolhe entre os valores oferecidos no conjunto de soluções viáveis a melhor solução para o problema.

- calculaR : PO x SL $\rightarrow$ V, calcula a função objetivo para a solução considerada (calculaR(I,s)=FO(I,s,tamanho(I)).

- F^* : PO $\rightarrow$ V t.q. $F^*(I)$:=FO(I,s,tamanho(I)), onde s=solução ótima de I.

- cardinalidade : SP $\rightarrow$ $\aleph$, t.q. cardinalidade(S) := cardinalidade do conjunto S.

Predicados Auxiliares

- Bemtransf: PO^2 x $\Re$ $\rightarrow$ {T,F}, Bemtransf(I,I',ξ):= $|(F^*(I) - F^*(I'))/F^*(I)| \leq \xi$

- MesmaSol: $PO^2 \rightarrow \{T,F\}$, MesmaSol(I,I') := $(\forall s)$ [SolOtima(I, s) $\Leftrightarrow$ SolOtima(I',s)].

- SolOtima: PO x SL $\rightarrow$ {T,F}, SolOtima(I,s) := s é solução ótima de I.

-SolAprox: PO x SL x $\Re \rightarrow$ {T,F}, SolAprox(I,s,ξ) := $|(F^*(I) - FO(I,s,tamanho(I)))/F^*(I)| \leq \xi$.

- LimiteInf : PO x $\Re \rightarrow$ {T,F}, LimiteInf(I,Lb) := $F^*(I) \geq Lb$.

- ContemSol : PO x SP x $\aleph \rightarrow$ {T,F}, ContemSol(I,S,k) := S contém uma solução ótima de I, restrita a k variáveis.

- Domina SL^2 x RD $\rightarrow$ {T,F}, Domina(s_1,s_2,Rd) := s_2 domina s_1 pela regra de dominância Rd.

- BemDef : PO x RD $\rightarrow$ {T,F}, BemDef(I,RD) := $(\forall s_1, s_2)$ { [s_1, s_2 $\in$ SL $\wedge$ Domina(s_1,s_2,RD) $\wedge$ SolOtima(I,s_2)] $\Rightarrow$ SolOtima(I, s_1)}.

- CardBoa : SP x $\aleph^2$ x $\Re \rightarrow$ {T,F}, CardBoa(S,k,n,ξ) := ($\exists c_1, c_2 \in \aleph$) (cardinalidade(S) $\leq c_1 + c_2.k.(n/\xi)$).

Axiomas : ACAX

AC1: $(\forall I)(\forall \xi)(\forall Lb)$ {LimiteInf(I,Lb) $\Rightarrow$ Bemtransf(I,transforma1 (I,ξ,Lb))}

AC2: $(\forall I)(\forall \xi)(\forall Lb)$ MesmaSol(I,transforma2(I,ξ,Lb))

AC3: $(\forall I)$ ContemSol(I,inicializa( ),0)

AC4: $(\forall I)(\forall S)(\forall k)(\forall Rd)$ {BemDef(I,Rd) $\Rightarrow$ [ContemSol(I,S,k) $\Rightarrow$ ContemSol(I,calculaS(I,S,Rd,k),k+1)]}

AC5: $(\forall I)(\forall S)(\forall \xi)$ { ContemSol (I,S,tamanho(I)) $\Rightarrow$ SolOtima(I, calculaSO (I,S))}

AC6: $(\forall I)(\forall I')(\forall I")(\forall \xi)(\forall s)$ {[Bemtransf(I,I',ξ) $\wedge$ MesmaSol(I',I") $\wedge$ SolOtima(I",s)] $\Rightarrow$ SolAprox(I,s,ξ)}

AC7: $(\forall I)(\forall \xi)$ CardBao(inicializa( ),0,tamanho(I),ξ)

AC8: $(\forall I)(\forall S)(\forall Rd)(\forall k)$ {BemDef(I,Rd) $\Rightarrow$ [CardBoa(S,k,tamanho(I),ξ) $\Rightarrow$ CardBoa(calculaS(I,S,Rd),k+1, tamanho(I),ξ)]}

Programa: ACERCAMENTO

entrada: I, ξ, Lb, Rd

1.I' ← transforma1 (I,ξ,Lb);

2.I" ← trasforma2 (I',ξ,Lb);

3.S ← inicializa ( );

4.n ← tamanho (I)

5.para k=0 até n-1 faça

6. S ← calculaS (I",S,Rd,k);

7.fim-para

8.s ← calculaSO (I",S);

9.v ← calculaR (I,s)

saída: s, v

Entradas:

- I ∈ PO :instância de entrada.

- ξ ∈ ℜ: requerimento de exatidão.

- Rd ∈ RD: regras de dominância.

Saídas:

- s ∈ SL: solução alcançada.

- v ∈ V: valor da função objetivo para a solução alcançada.

A prova da correção do programa ACERCAMENTO pode ser extensa e cansativa, mas não é difícil e pode ser encontrada no Apêndice E1 de [TOS 88].

5.UMA IMPLEMENTAÇÃO

Uma implementação de um tipo abstrato é uma definição do tipo em um nível de abstração mais baixo. Uma implementação do tipo definido na secção anterior pode ser uma definição algorítmica das funções transforma1, transforma2 e calculaS.

Definição de transforma1

Programa: transforma1

entrada: (P= $(p_1,\ldots,p_n)$, A, B), ξ, Lb

1. n ← tamanho(P)
2. para i=1 até n faça
3. $q_i \leftarrow \left\lfloor \frac{p_i . n}{Lb . \xi} \right\rfloor \frac{Lb . \xi}{n}$
4. fim-para
5. Q ← $(q_1, q_2, \ldots, q_n)$

saída: (Q,A,B)

Definição de transforma2

Programa: transforma2

entrada: (P= $(p_1,\ldots p_n)$, A, B), ξ, Lb

1. n ← tamanho(p)
2. para i=1 até n faça
3. $q_i \leftarrow \frac{p_i . n}{Lb . \xi}$
4. fim-para
5. Q ← $(q_1,q_2,\ldots,q_n)$

saída:(Q, A, B)

Definição de calculaS

$SS = \bigcup_{S \in SP} S$ = conjunto de todas i-uplas $1 \leq i \leq n$, soluções parciais

Funções e Predicados Auxiliares

- completa: SS x $\mathbb{N}$ ← SS, completa($(x_1,x_2,\ldots,x_k)$,k) considera a k-ésima variável do problema resultando em uma solução possível com mais um elemento, $(x_1,x_2,\ldots,x_k,1)$.

- Solviável SS x SP x PO x $\aleph \rightarrow \{T,F\}$, Solviável(s,S,I,k) verifica se a solução parcial s é viável para k variáveis.

- restringe: SP x RD x PO $\rightarrow$ SP, restringe(S,Rd,I) restringe o conjunto de soluções parciais S de I, de maneira que o novo conjunto de soluções parciais não contenha duas soluções s_1, s_2, com s_1 dominando s_2 (dominando de acordo com as regras de dominância Rd).

- Domina: SS^2 x RD $\rightarrow$ {T,F}, Domina(s_1, s_2,Rd) verifica se s_1 domina s_2 de acordo com as regras de dominância Rd.

- Éviável: PO x SL $\rightarrow$ {T,F}, Éviável(I,s)= $\sum_{i=1}^{n} a_{ij}\, s_i \leq b_j$, onde n=tamanho(I), I=(P,A,B), A = (a_{ij}), B =$(b_1, b_2, \ldots, b_n)$ e s= $(s_1, \ldots, s_n)$.

Programa: calculaS

{φ(I,S,Rd,k) := BemDef(I,Rd) $\wedge$ ContemSol(I,S,k)}

entrada: I, S, Rd, k

1. S' $\leftarrow \phi$
2. para s $\in$ S faça
3. s' $\leftarrow$ completa(s,k)
4. se Solviável(s',S,I,k+1) então S'$\leftarrow$ S' $\cup$ {s'}
5. fim-para

{invariante: ContemSol(I,S',k+1) $\wedge$ BemDef(I,Rd)}

6. S $\leftarrow$ restringe(S',Rd,I)

saída: S

{ψ(I,S,Rd,k) = ContemSol(I,S,k+1)}

Axiomas: CalSAX

ACS1. $(\forall I)(\forall S)(\forall s)(\forall k)$ {[ContemSol(I,S,k) $\wedge$ S' = S $\cup$ {completa(s,k)} $\wedge$ Solviável(completa(s,I),S,I)] $\Rightarrow$ ContemSol(I,S',k+1)}

ACS2. $(\forall I)(\forall S)(\forall k)(\forall Rd)\{[BemDef(I,Rd) \wedge ContemSol(I,S,k)] \Rightarrow [ContemSol(I,restringe(S,Rd,I),k) \wedge (\forall s_1, s_2 \in restringe(S,Rd,I))(\neg Domina(s_1,s_2,Rd))]\}$

ACS3. $(\forall I)(\forall S)(\forall S')(\forall k)\{ContemSol(I,S,k)\ [(\forall s \in S)\ Solviável(completa(s,k),S,I,k+1) \Rightarrow completa(s,k) \in S'] \Rightarrow ContemSol(I,S',k+1)\}$

Note que calculaS, diferente de trasforma1 e transforma2, contém funções abstratas cujas semânticas precisam ser definidas e são definidas pelos axiomas CalSAX. Um novo módulo de implementação pode definir as funções abstratas de calculaS (completa e restringe).

A correção da implementação é fácilmente provada ([TOS 88], secção 6.2.2).

6. EXEMPLO

Este exemplo foi retirado de [HOR 78] e é a solução de uma instância do problema da mochila.

O problema da mochila é uma variante do problema de maximização estudado e pode ser definido assim : calcular

$$\max_x \sum_{i=1}^{n} p_i\, x_i \quad \text{restrito a} \quad \sum_{i=1}^{n} w_i\, x_i \leq M,\ x_i=0 \text{ ou } 1,\ 1 \leq i \leq n.$$

Chame $r=\sum_{j=1}^{i} p_j\, x_j$ e $t=\sum_{j=1}^{i} w_j\, x_j$, (r,t) são as atribuições possíveis. A regra de dominância é: (r_1,t_1) domina (r_2,t_2) sss $t_1 \leq t_2$ e $r_1 \geq r_2$.

A instância é I=(P,W,M), onde $P=(p_1,p_2,p_3,p_4,p_5)=W=(w_1,w_2,w_3,w_4,w_5)=(1,\ 2,\ 10,\ 100,\ 1000)$, M=1112. Para esta instância r=t.

Em $V^{(i)}$ serão representados os valores da função objetivo para as soluções parciais viáveis ao invés das próprias soluções parciais viáveis.

$V^{(0)}=\{0\}$,

$V^{(1)}=\{0,1\}$,

$V^{(2)}=\{0,1,2,3\}$,

$V^{(3)}=\{0,1,2,3,10,11,12,13\}$,

$V^{(4)}=\{0,1,2,3,10,11,12,13,100,101,102,103,110,111,112,113\}$,

$V^{(5)}=\{0,1,2,3,10,11,12,13,100,101,102,103,110,111,112,113,$
$1000,1001,1002,1003,1010,1011,1012,1013,1100,1101,1102,$
$1103,1110,1111,1112\}$.

A solução ótima é (0,1,1,1,1) com valor ótimo 1112. Usando o método de Acercamento, com $\xi=(1/10)$, $Lb=\max\{p_i\}=1000$, tem-se $I"=(Q,W,M)$, $Q=(0,0,0,5,50)$, $q_i=\lfloor\frac{p_i.n}{Lb.\xi}\rfloor=\lfloor\frac{p_i}{20}\rfloor$.

$V^{(0)}=V^{(1)}=V^{(2)}=V^{(3)}=\{(0,0)\}$ (Obs.: (0,1) não está em $V^{(3)}$ por que é dominado por (0,0)).

$V^{(4)}=\{(0,0),(5,100)\}$, $V^{(5)}=\{(0,0),(5,100),(50,1000),(55,1100)\}$.

A solução ótima em I" é s=(0,0,0,1,1), com valor FO(I",s,5)=55. s transportada para I resulta FO(I,s,5)= 1100. $(F^*(I) - FO(I,s,5))/F^*(I)=(1112-1100)/1112=12/1112 < 0.011 < \xi$.

7.CONCLUSÃO

Muitas vezes os dados de um problema são aproximados, assim uma solução aproximada "suficientemente próxima" da solução exata é tão significativa quanto a própria solução exata. Baseados nesse fato foram desenvolvidos métodos de solução de problemas que encontram soluções aproximadas, isto é, algoritmos aproximativos. A condição "suficientemente próxima", para muitas aplicações é uma exigência de qualidade mínima para aceitação de uma solução aproximada. Esta exigência é posta neste texto como requerimento de exatidão.

Métodos de desenvolvimento de algoritmos como Divisão e Conquista e Programação Dinâmica já foram exaustivamente estudados [VEL 80], [TOS 85], [TOS 86], [TOS 88]. Métodos para desenvolvimento de algoritmos aproximativos são mais complexos, já que os algoritmos gerados tem que satisfazer um certo requerimento de exatidão e de complexidade, pois um algoritmo aproximativo para ser útil tem que dar uma solução satisfatóriamente próxima da solução exata em um tempo significativamente menor que o algoritmo exato.

A especificação formal do método de Acercamento pretende ser uma contribuição no sentido de aumentar o conhecimento sobre o método, melhorando a comprensão do mesmo e facilitando a sua

utilização. Além disso, a especificação formal é necessária para um estudo teórico do método, como por exemplo a análise comparativa entre mdaa's.

REFERÊNCIAS

[HOR 78] HOROWITZ, E. & SAHNI, S. "Fundamentals of Computer Algorithms". Potomac, Md. Comp. Sci. Press 1978.

[TOS 85] TOSCANI,L. V. & VELOSO, P. A. S. Uma Especificação Formal para a Programação Dinâmica. In: SEMINÁRIO INTEGRADO DE SOFTWARE E HARDWARE, 12., Porto Alegre,20-27/jul, 1985. Anais. Porto Alegre, SBC/CLEI/UFRGS, 1985. p. 477-86.

[TOS 86] TOSCANI,L. V. & VELOSO,P.A.S. Divisão e Conquista: análise da complexidade. In: SEMINÁRIO INTEGRADO DE SOFTWARE E HARDWARE, 13, Olinda, 19-25/jul.,1986. Anais. Recife SBC,1986. p.89-104.

[TOS 88] TOSCANI, L. V. MÉTODOS DE DESENVOLVIMENTO DE ALGORITMOS : ANÁLISE COMPARATIVA E DE COMPLEXIDADE Tese de doutorado. Rio de Janeiro, Depto. de Informática, Puc/RJ.1988.

[VEL 80] VELOSO, P.A.S."Divide-and-Conquer via data types" In:LATIN AMERICAN CONFERENCE ON INFORMATICS, 7, Caracas, 1980. Proceedings.