Heuristické optimalizačné procesy

Únik z lokálneho optima

Marian.Mach@tuke.sk
http://neuron.tuke.sk/~machm

Marec, 2013

Únik z lokálneho optima

- Prístupy
  - opakované reštarty
  - zväčšenie okolia
    - prepínanie okolí rôznych veľkostí
  - dlhý krok mimo okolia
  - akceptácia horšieho riešenia
  - pamäť
  - dynamické lokálne prehľadávanie

Opakované reštarty

- Po dosiahnutí lokálneho optima je algoritmus reinicializovaný
- vhodná stratégia ak:
  - počet lokálnych optím nie je príliš veľký
    - pri optimalizačnom probléme vhodné aj pri väčšom počte lokálnych optím
  - cena reštartu nie je príliš nákladná

Štruktúra II s reštartom

- input: π
- output: $r \in S \mid \square$
- r = □, g(r) = -∞

then

- i = 1
- **repeat**
- s = urp( )
- **while**(#l(s) > 0)
- s = select(l(s))
- **endwhile**
- **if**(g(s) > g(r)) **then**
- r = s
- **endif**
- i = i + 1
- **until** i > max

if(valid(r))

return r

else

return □

endif

Väčšie okolie

- Zväčšenie okolia
  - viac kandidátov v okolí (väčšia šanca na lepšieho kandidáta)
  - graf okolia s menším priemerom
  - lokálne optimum malého okolia nemusí byť optimom okolia väčšieho
  - rastú nároky na čas potrebný pre preskúmanie okolia
- modifikácie
  - orezávanie okolia
  - pivotné pravidlo (funkcia *step*)

Pivotné pravidlo

- $I(s) = \{ x \in N(s) \mid g(x) > g(s) \}$
- $I^*(s) = \{ x \in N(s) \mid g(x) = max \{ g(y) \mid y \in N(s) \} \}$

- Najlepšie zlepšenie
  - $p(x) = 1 / \#I^*(s)$, $x \in I^*(s)$; inak $p(x) = 0$
- Prvé zlepšenie
  - $p(x_i) = 1$, $x_i \in I(s)$, $\neg(x_{i-j} \in I(s))$
- Náhodné prvé zlepšenie
- Náhodné zlepšenie
  - $p(x) = 1 / \#I(s)$, $x \in I(s)$; inak $p(x) = 0$

Prepínanie okolí

- Používané okolia:
  - malé okolie (zlepšovanie aktuálneho kandidáta)
  - veľké okolie (zabránenie uviaznutiu)
- prepínanie okolí – VNS algoritmy
  - jedna z možností je VND
    - *k* okolí: $N_1 < N_2 < \ldots < N_k$
    - *zväčšovanie okolia:* $N_i \rightarrow N_{i+1}$
    - *zmenšovanie okolia:* $N_i \rightarrow N_1$
    - *začiatok:* N_1
    - *koniec: uviaznutie v* N_k

Štruktúra VND

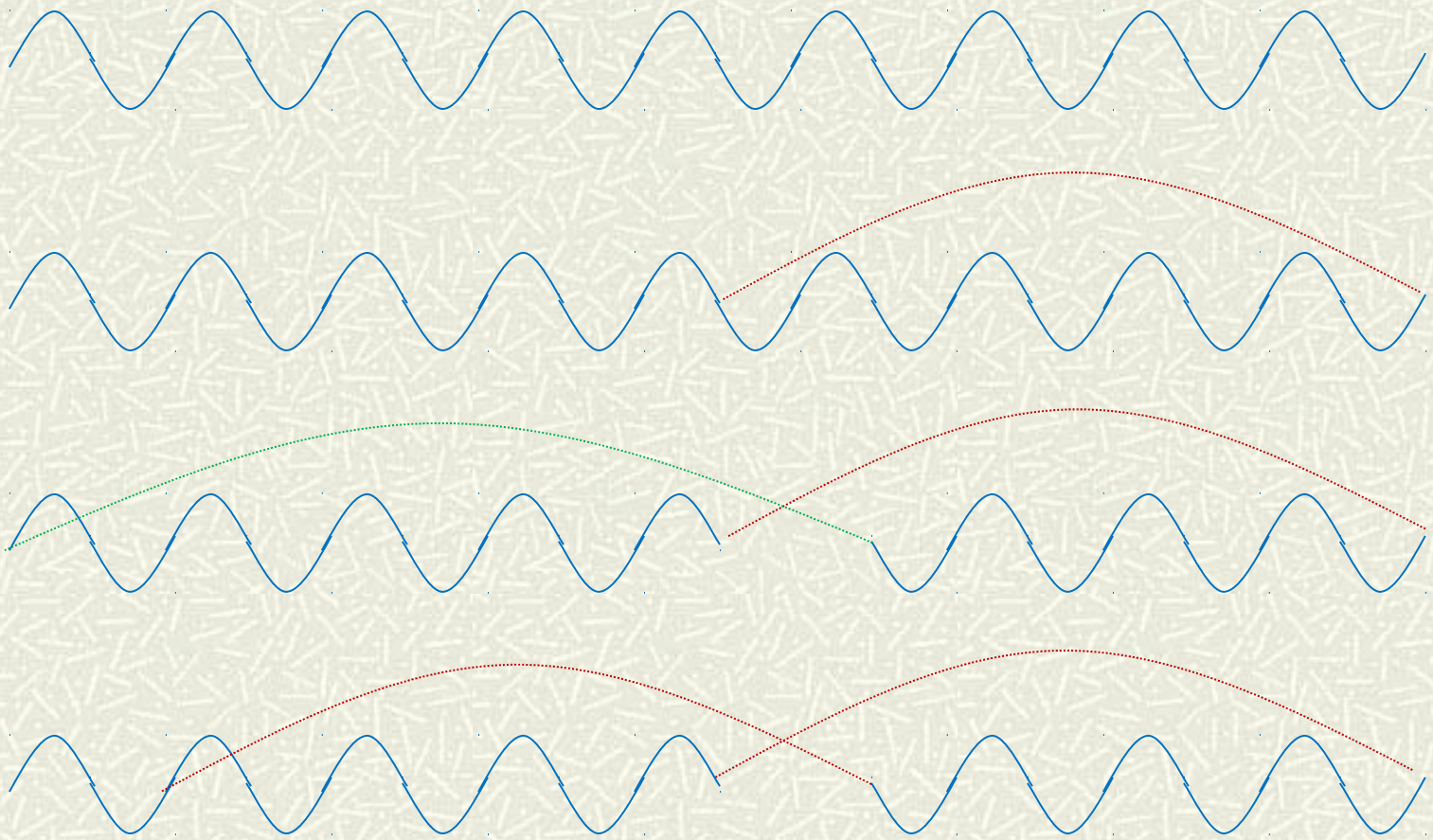
- input: π
- output: $r \epsilon S$ | □
- r = urp()
- i = 1
- **repeat**
- s = best(N_i(r))
- **if**(g(s) > g(r)) **then**
- r = s
- i = 1
- **else**
- i = i + 1
- **endif**
- **until** i > k

if(valid(r)) **then**
return r
else
return □

Dlhý krok

- Kompozícia dlhého kroku z viacerých krokov v malom jednoduchom okolí
  - kroky pozostávajú z rôzne dlhých sekvencií jednoduchých krokov
  - sekvencie spĺňajú ohraničenia na prijateľnosť
    - reštrikcia ceny
    - tabu reštrikcia
- algoritmy VDS
  - príkladom je Lin-Kernighan algoritmus (TSP)
    - sekvencia 2-exchange krokov

LK – konštrukcia kroku



Štruktúra VDS

```
input: π
output: r ϵ S | □
        r = urp()
        repeat
                t = r
                repeat
                        s = t
                        t = bestfeasible( N(s) )
                until termconstruct( t, s )
                impr = not
                if g( r ) < g ( s ) then
                        r = s
                        impr = yes
                endif
        until not impr
if( valid( r ) ) then
        return r
else
        return □
```

Akceptácia horšieho riešenia

- Horší kandidát
  - jediná možnosť (uviaznutie v lokálnom optime)
  - jedna z možností (existuje aj lepší kandidát)
- stratégia výberu zhoršujúceho kroku
  - iná možnosť nie je
  - pravidelná alternácia
  - pravdepodobnostný výber
- pravdepodobnosť zhoršujúceho kroku
    - pevná (RII)
    - adaptívna (PII)

Randomizované iteračné zlepšovanie

- Funkcia step
  - zlepšujúci krok – $step_{ii}$
  - zhoršujúci krok – $step_{urw}$
  - parameter *wp*

    $step = wp * step_{urw} + (1 - wp) * step_{ii}$
- zmena terminačného kritéria
  - dosiahnutie limitu
  - absencia zlepšenia
- únik z lokálneho extrému
- dosiahnutie globálneho optima

Štruktúra RII

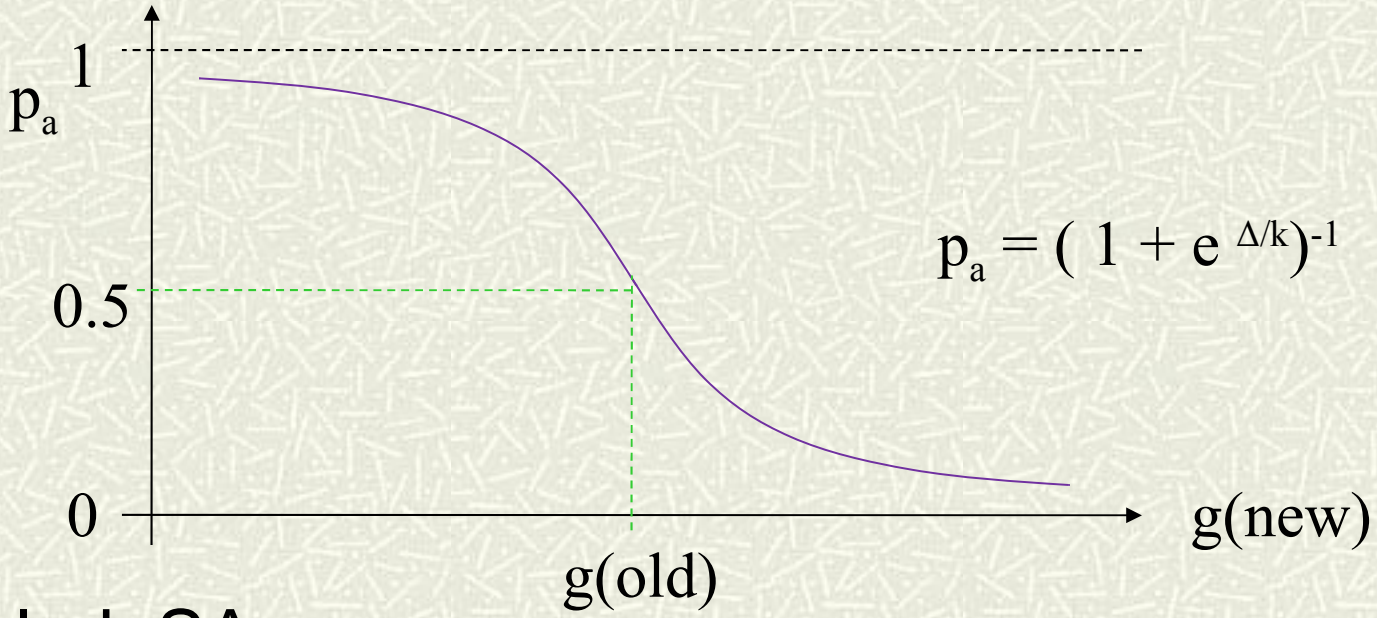
- input: π
- output: $r \in S \mid \square$

```
        s = urp()
        r = s
        repeat
                if( rand( 0-1 ) < wp ) then
                        s = step_urw( s )
                else
                        s = step_ii( s )
                endif
                if ( g( s ) > g( r ) ) then
                        r = s
                endif
        until term( s )
```

```
if( valid( r ) ) then
        return r
else
        return □
```

Pravdepodobnostné iteračné zlepšovanie

- Funkcia *step* – dvojkrokový proces
  - výber kandidáta
    - *p(x) = 1 / #N(s), x ϵ N(s)*
  - akceptovanie kandidáta



- príklad: SA

Štruktúra PII

- input: π
- output: $r \epsilon S$ | □
- s = urp()
- r = s
- **repeat**
  - s' = urw(N(s))
  - $p_a = f_{akc}(s')$
  - **if**(rand(0-1) < p_a) **then**
    - s = s'
    - **if** (g(s) > g(r)) **then**
      - r = s
    - **endif**
  - **endif**
- **until** term(s)

if(valid(r)) **then**
  return r
else
  return □

Zakázané lokálne prehľadávanie

- Funkcia *step* – najlepšie zlepšenie
  - *p(x) = 1 / #I*(s), x ϵ I*(s)-s;* inak *p(x) = 0*
- krátkodobá pamäť znemožňuje návrat
  - zabránenie (posledne) skúmaných kandidátov
  - dynamická reštrikcia okolia
- obsah pamäti
  - reverzie predchádzajúcich transformácií
  - iba dočasné uchovávanie obsahu
  - reaktívne ZP
- realizácia krátkodobej pamäte

Rozšírenia zakázaného prehľadávania

- Ašpiračné kritérium
  - podmienka ignorovania krátkodobej pamäte
- dlhodobá pamäť
  - cieľom je rovnomerné používanie transformácií
  - frekvencie použitia transformácií
  - výber transformácie
    - $g(t_i(s))$ vers. $g(t_i(s)) + k*DP(t_i)$
  - použitie dlhodobej pamäte

Štruktúra TS

- input: π
- output: *r* ϵ *S* | □

```
        s  = urp()
        r = s
        repeat
                s' = ac( N( s ) )
                if(  s'  ) then
                        s = s'
                else
                        s = step( restr( N( s ), M ) )
                endif
                update-memory( M, s )
                if ( g( s ) > g( r ) ) then
                        r = s
                endif
        until term( s )
```

```
if( valid( r ) ) then
        return r
else
        return □
```

Dynamické lokálne prehľadávanie

- penalizácia lokálneho optima vedúca na degradáciu ohodnocovacej funkcie *g*
  - opakovanie až kým kandidát prestane byť lokálnym optimom
- penalizácia
  - stratégia penalizácie
    - adaptívna/multiplikačná schéma, parametrizácia
  - voľba komponentov lokálneho optima
    - $g'(s) = g(s) + \sum_{i \in sc} pen(i)$
  - voľba komponentu s najväčšou užitočnosťou
    - $util(i,s) = f_i(s) / (1 + pen(i))$

Štruktúra DLS

- input: π
- output: $r \in S$ | □
- s = urp()
- $s = LS_{II}(s)$
- r = s
- **repeat**
- g = update-penalties(g, s)
- $s = LS_{II}(s)$
- **if** ($g_{orig}(s) > g_{orig}(r)$) **then**
- r = s
- **endif**
- **until** term(s)

if(valid(r)) **then**
return r
else
return □