ALGORITEM OHLAJANJE (SIMULATED ANNEALING)

INFORMATICA 2/91

Keywords: combinatorial optimization, probabilistic algorithm, Simulated Annealing, randomized local search

Janez Žerovnik
Inštitut za matematiko, fiziko in mehaniko
Jadranska 19, 61111 Ljubljana
Slovenia, Yugoslavia

Povzetek: V preglednem sestavku opišemo popularni algoritem za reševanje splošnega problema kombinatorične optimizacije, ki ga imenujemo Ohlajanje (angl. Simulated Annealing, oznaka $\mathcal{SA}$). Sledi pregled literature in izrek o asimptotskem obnašanju algoritma v primerjavi s preprostim algoritmom, inačico algoritma lokalna optimizacija ($\mathcal{RLS}$). Posplošitev izreka velja tudi za posplošeni algoritem, ki pokriva vse znane paralelizacije algoritma $\mathcal{SA}$.

Abstract: ALGORITHM SIMULATED ANNEALING. The algorithm Simulated Annealing ($\mathcal{SA}$ for short) for solving the general problem of combinatorial optimization has received considerable attention in the literature in the last years. A survey of a number of both theoretical and practical results is given. A theorem, which shows that the asymptotic behavior of the Simulated Annealing algorithm is worse than the asymptotic behavior of the randomised local search algorithm ($\mathcal{RLS}$ for short), is discussed [Žero88]. A generalization of the theorem holds for a very general algorithm which covers many known paralelizations of the $\mathcal{SA}$ algorithm [BFZ89a].

1. Uvod

V teoretičnem računalništvu (ali algoritmiki, kot bi tej veji matematike rekel Knuth [Knut81]) je prevladalo mnenje, da intuitivnima pojmoma lahkih in težkih problemov ustrezata razreda P in NP. V razredu P so problemi, za katere obstajajo polinomski algoritmi. Število korakov, ki jih polinomski algoritem potrebuje za rešitev problema, ki ga lahko opišemo s podatki velikosti n (bitov, besed ali cifer) je omejeno z neko polinomsko funkcijo n-ja. V razredu NP so problemi, za katere obstajajo polinomski algoritmi, ki za dani problem in dano rešitev preverijo, ali je rešitev ustrezna. Očitno torej velja P$\subseteq$NP. V razredu NP je posebej odlikovana skupina NP-polnih problemov. Za NP-polne probleme velja naslednje: če bi obstajal polinomski algoritem za reševanje enega od njih, potem bi obstajali polinomski algoritmi za vse probleme iz razreda NP. Krajše bi to zapisali P=NP. Vprašanje, ali je P$\neq$NP je verjetno najbolj znan odprti problem teoretičnega računalništva. Po skoraj dvajsetih letih brez odgovora vedno bolj prevladuje mnenje, da je verjetno razred P prava podmnožica razreda NP. Za uvodno branje o časovni zahtevnosti algoritmov je še vedno odličen vir že klasična knjiga [GaJo79], v Slovenščini pa na primer [HoUl86] ali [KLPP86].

Najboljši znani algoritmi za NP-polne probleme so eksponentni, zanje je število potrebnih korakov v najslabšem primeru navzdol omejeno z eksponentno funkcijo velikosti problema. Zaradi hitre rasti eksponentnih funkcij to običajno pomeni, da ne znamo dobiti točnih rešitev že za sorazmerno majhne velikosti problemov. Če se sprijaznimo s predpostavko P$\neq$NP, potem je smiselno raziskovati algoritme, ki bodo v razumno kratkem času dajali rešitve, ki bodo ustrezali določenim zahtevam. Aproksimacijski algoritmi na primer dajejo rešitve, za katere je znano, koliko največ utegnejo odstopati od optimalne rešitve. V zadnjem času so sorazmerno popularni verjetnostni algoritmi. Algoritmu rečemo *verjetnostni*, če nekateri koraki niso vnaprej natanko določeni s podatki, in rezultat pri istih podatkih torej ni nujno enak pri vsakem teku algoritma. Članki verjetnostnih algoritmih so na primer [Wels83, Maff86, Frie79, Rabi76]. Pri verjetnostnih algoritmih imamo lahko različne garancije kvalitete, na primer:

- algoritme, ki vedno dajo optimalno rešitev in imajo v povprečju polinomsko časovno zahtevnost,
- algoritme s polinomsko časovno zahtevnostjo, ki dovolj pogosto dajo dobre rezultate,
- polinomske algoritme z omejeno verjetnostjo napake pri negativnem odgovoru in zanesljivim (verjetnost napake 0) pozitivnim odgovorom,
- algoritme, pri katerih se verjetnost izračuna optimalne rešitve s časom približuje 1.

Med verjetnostnimi algoritmi je bil v zadnjem času deležen precej pozornosti algoritem $\mathcal{SA}$. Samo v tem članku med literaturo navajamo preko 30 referenc o tem algoritmu, bibliografija pa s tem še zdaleč ni izčrpana, saj se članki na to temo še vedno pojavljajo [Sasa91]. Za ilustracijo omenimo, da v oglasu za knjigo [John90] omenjajo preko 300 člankov o $\mathcal{SA}$! O popularnosti priča tudi veliko število imen, ki jih navajajo v uvodu ene od knjig o tem algoritmu [LaAr87]: Monte Carlo annealing, statistical cooling, probabilistic hill climbing, stochastic relaxation ali probabilistic exchange algorithm. Leta 1953 je Metropolis s sodelavci [MRRT53] uporabil računalnik za simulacijo fizikalnega modela ohlajanja snovi. Kasneje so Kirckpatrick s sodelavci [KiGV83] in neodvisno Černy (Czerny) [Čern84] opazili analogijo med reševanjem problema kombinatorične optimizacije in Metropolisovo simulacijo fizikalnega modela. Verjetno je prav zaradi duhovite analogije algoritem hitro postal široko poznan in nekoliko 'moderen'. Večina

avtorjev kot vir za $\mathcal{SA}$ citira članek [KiGV83], zaradi pravičnosti pa velja omeniti, da so analogijo med statistično mehaniko in optimizacijo (zveznih) funkcij opazili že prej Khačaturjan, Semenovskaja, Vainstein [KhSV81] in Pincus [Pinc70], vendar sta članka ostala brez širšega odmeva. V nadaljevanju bomo povzeli izrek, da algoritem 'konvergira'. Točneje: pri določenih pogojih (dovolj počasno ohlajanje) za pripadajočo (časovno) nehomogeno Markovsko verigo obstaja limitna porazdelitev, v kateri imajo pozitivno verjetnost natanko globalno optimalna stanja (optimalne rešitve). V praksi je ohlajanje seveda vedno 'prehitro'. V velikem številu poročil o poskusih (glej pregled v nadaljevanju) poročajo o dobrih rezultatih. To sicer včasih pomeni, da so dobljene rešitve dotlej najboljše znane za nekatere referenčne primerke nekaterih problemov, vendar gre to vedno na račun zelo velikega računskega časa. V literaturi najdemo celo vrsto teoretičnih analiz algoritma. Tukaj ob knjigah [LaAr87, Laar88] in članku [MiRS86], po katerem povzemamo izreke o konvergenci algoritma, omenimo še enega. Sasaki in Hajek v [SaHa88] pokažeta, da je za problem maksimalnega prirejanja (angl. maximum matching) cel razred algoritmov tipa $\mathcal{SA}$ gotovo več kot polinomske časovne zahtevnosti.
Algoritma $\mathcal{SA}$ in $\mathcal{RLS}$ sta definirana (glej razdelka 4. in 2.) dovolj splošno, da ju lahko uporabimo za reševanje kateregakoli problema kombinatorične optimizacije, zato ju je smiselno primerjati. Glavna zamera algoritmu $\mathcal{RLS}$ (oziroma notranji zanki algoritma) je to, da se 'ujame' v lokalnih optimumih. Ko začnemo z novo začetno rešitvijo v nekem smislu zavržemo vso informacijo, ki smo jo prigarali z dotedanjim računanjem (razen seveda dotlej najboljše dobljene rešitve). Če dovolimo tudi korake 'navzgor' (v smeri povečanja stroškovne funkcije) se algoritem ne ustavi več v lokalnih optimumih. Tako dobimo takoimenovane 'plezalne' (angl. hill climbing) algoritme, med katere uvrščamo tudi algoritem $\mathcal{SA}$. Na račun možnosti plezanja navzgor pa se sicer navidezno še vedno zelo enostavni algoritem precej zaplete, saj je potrebno poiskati prave vrednosti parametrov, ki kontrolirajo, kdaj in koliko dovolimo korake 'navzgor'. Pri algoritmu $\mathcal{SA}$ kontrolnemu parametru običajno rečemo temperatura. Problem, kako temperaturo zniževati, da bomo dobili dobre rezultate, ni enostaven in vse kaže, da je odgovor precej odvisen tudi od tipa problema, ki ga želimo reševati.
Primerjava algoritmov $\mathcal{SA}$ in $\mathcal{RLS}$ je po [LaAr87, Laar88] odprt problem. V nadaljevanju bomo pregledali nekaj poročil o eksperimentih z obema algoritmoma, ki ne dajo enoznačnega odgovora na gornje vprašanje. Tu bomo pokazali (Izrek 2), da je algoritem $\mathcal{SA}$ asimptotsko slabši od lokalne optimizacije in tako deloma odgovorili na gornje vprašanje.

2. Splošni problem kombinatorične optimizacije

Mnoge diskretne optimizacijske probleme lahko gledamo kot posebne primere *splošnega problema kombinatorične optimizacije*, ki ga bomo definirali v tem razdelku. V naslednjem razdelku bomo dodali še konkretni primer, problem trgovskega potnika.
Dana naj bo (končna ali števna) množica objektov $\mathbf{S}$. Elementom $\mathbf{S}$ rečemo *dopustne rešitve* ali *stanja*. Množica $\mathbf{S}$ je lahko podana opisno, včasih pa se moramo zadovoljiti s tem, da je dan algoritem, ki konstruira elemente iz $\mathbf{S}$. V obeh primerih bomo privzeli, da obstaja verjetnostni algoritem $\mathcal{R}$, ki generira elemente iz $\mathbf{S}$ in ima naslednjo lastnost: za poljubni objekt $o \in \mathbf{S}$ je verjetnost dogodka, da je $\mathcal{R}$ zgeneriral ravno o, pozitivna. Formalno

$$P\{\mathcal{R} = o\} = p_o \geq \min_{o \in \mathbf{S}} p_o > 0. \qquad (1)$$

Opomba: če za trenutek pozabimo na (daljši ali krajši) računski čas, lahko rečemo, da je $\mathcal{R}$ slučajna spremenljivka z zalogo vrednosti $\mathbf{S}$.
Lep posebni primer dobimo, če je množica $\mathbf{S}$ končna in če $\mathcal{R}$ generira vse elemente množice $\mathbf{S}$ z enako verjetnostjo.
Na množici $\mathbf{S}$ je definirana *stroškovna funkcija* $c : \mathbf{S} \rightarrow \mathbf{C}$. Za množico vrednosti $\mathbf{C}$ lahko vzamemo na primer množico naravnih $\mathbb{N}$ ali realnih števil $\mathbb{R}$.
Naloga je poiskati minimum stroškovne funkcije c na množici objektov $\mathbf{S}$.
Povzemimo:

Definicija 1 *Splošni problem kombinatorične optimizacije je družina primerkov. Primerek (splošnega) problema kombinatorične optimizacije je par* $(\mathbf{S}, c)$, *kjer je* $\mathbf{S}$ *množica (množica dopustnih rešitev),* c *pa je preslikava* $S \rightarrow \mathbf{C}$ *(stroškovna funkcija). Treba je poiskati tak* $o_{min} \in \mathbf{S}$, *da velja*

$$c_{min} = c(o_{min}) = \min_{o \in \mathbf{S}} c(o) \qquad (2)$$

Problem iskanja maksimuma definiramo analogno, je pa trivialno ekvivalenten problemu minimuma na isti množici $\mathbf{S}$ s stroškovno funkcijo $\tilde{c} = -c$.
V [KLPP86] pravkar definirane pojme imenujejo nekoliko drugače. Primerku problema rečejo naloga, stroškovna funkcija pa je namenska ali ciljna funkcija.
V tem delu se bomo omejili na probleme s končno množico $\mathbf{S}$. Čeprav končna, pa je množica $\mathbf{S}$ pogosto zelo obsežna (recimo velikosti, ki je eksponentna funkcija velikosti problema). V primeru trgovskega potnika je kar $n!$ permutacij mest, če je vsak par mest povezan.
Zaradi obsežnosti pregledovanje vse množice $\mathbf{S}$ običajno ne pride v poštev. Pri NP–polnih problemih je pregled množice $\mathbf{S}$ algoritem eksponentnega reda. Doslej še nikomur ni uspelo odkriti algoritma, ki bi bil bistveno hitrejši (ki bi imel časovno zahtevnost manj kot eksponentnega reda). Ker verjetno velja hipoteza P≠NP, je za to tudi kaj malo upanja.
Pri obravnavi splošnega problema kombinatorične optimizacije $(\mathbf{S}, c)$ bomo privzeli, da obstaja vsaj en verjetnostni algoritem $\mathcal{R}$ $(\mathcal{R} : \emptyset \rightarrow \mathbf{S})$ in vsaj en verjetnostni algoritem $\mathcal{N}$ $(\mathcal{N} : \mathbf{S} \rightarrow \mathbf{S})$. Verjetnostni algoritem V:$A \rightarrow B$ v našem primeru realizira neko preslikavo. Rezultat je element ciljne množice B, porazdelitev verjetnosti, da bo izračunan konkretni element pa je v splošnem odvisna od podatka, elementa množice A.
Množico $\mathbf{N}(x) = \{y | P(\mathcal{N}(x) = y) > 0\}$ imenujemo *soseščina* točke $x \in \mathbf{S}$. Z algoritmom $\mathcal{N}$ na naravni način vpeljemo razdaljo na množico $\mathbf{S}$, ki inducira neko topologijo. Pogosto je za dani problem mogoče na več načinov smiselno definirati pojem soseščine. V takih primerih nam različne izbire v splošnem dajo različne topologije na množici $\mathbf{S}$, s tem pa tudi različno obnašanje splošnih algoritmov, ki jih bomo definirali v nadaljevanju. Nas bo tu poleg topologije seveda zanimala tudi učinkovitost (hitrost) algoritma $\mathcal{N}$. Običajno lahko pričakujemo, da bomo imeli ali hiter algoritem $\mathcal{N}$ in "grdo" topologijo (z mnogimi lokalnimi ekstremi) ali pa obratno, "lepo" topologijo, vendar počasni algoritem $\mathcal{N}$. Za primerjavo različnih splošnih algoritmov

je pomembno tudi razmerje med časovnima zahtevnostima algoritmov $\mathcal{N}$ in $\mathcal{R}$.
Brž ko privzamemo, da imamo dan algoritem $\mathcal{N}$ in z njim topologijo, lahko definiramo *lokalni minimum* na S:

o_l je lokalni minimum, če velja $c(o_l) < c(o) \quad \forall o \in N(o) - \{o_l\}$

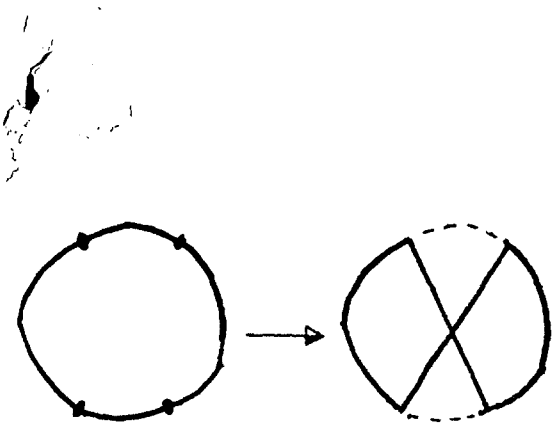
Analogno lahko definiramo *lokalni maksimum*.
Za reševanje splošnega problema kombinatorične optimizacije je znan enostavni algoritem, ki ga bomo imenovali algoritem lokalna optimizacija (angl. randomised local search $\mathcal{RLS}$ [Maff86], neighborhood search, iterative improvement [LaAr87, Laar88], multistart algorithm [Gida85]).

Algoritem $\mathcal{RLS}$: (lokalna optimizacija)
repeat
  generiraj slučajno začetno stanje
  repeat
    poišči soseda
    if je boljši **then** premakni se
  until ni boljšega soseda
until preveč korakov

Enostaven premislek nas pripelje do ugotovitve, da notranja **repeat until** zanka konča v lokalnih minimumih. V nobenem primeru v notranji zanki algoritem ne more iz lokalnega minimuma. To, da se 'ustavi' v lokalnih ekstremih, je tudi glavna zamera, ki jo temu algoritmu namenjajo kritiki [Laar88]. Argument je tembolj tehten, kolikor je $\mathcal{N}$ učinkovitejši od algoritma $\mathcal{R}$.
O časovni zahtevnosti tako splošnega algoritma se ne da povedati veliko. Prav mogoče je, da je celo samo eno iskanje lokalnega optimuma potreben nepolinomski računski čas. Zadostuje, da so okolice dovolj 'bogate'. O tem nas prepriča trivialni primer: vedno je mogoče definirati soseščino tako, da so vse dopustne rešitve dosegljive ena iz druge. Za $\mathcal{N}$ vzamemo kar $\mathcal{R}$ iz definicije problema kombinatorične optimizacije. V tem primeru potrebujemo za pregled ene same okolice $O(|S|)$ časa! (Za tolažbo pa bomo optimalno rešitev zanesljivo dobili.)
Zanimivo je, da celo za znano '2-opt' soseščino pri problemu trgovskega potnika časovna zahtevnost ene lokalne optimizacije ni znana [LITT89]. Menda je Kern [Kern86] nedavno za nekatere probleme trgovskega potnika pokazal, da je '2-opt' v povprečju polinomske časovne zahtevnosti.
Pregled različnih posplošitev algoritma $\mathcal{RLS}$ najdemo v [Maff86], kjer med drugim tudi algoritem Ohlajanje ($\mathcal{SA}$) definirajo kot posplošitev algoritma $\mathcal{RLS}$.

3. Primer problema kombinatorične optimizacije

Mnogi diskretni optimizacijski problemi se dajo zapisati v obliki problema kombinatorične optimizacije, bogato zalogo problemov najdemo na primer v [KLPP86, Koza86], verjetno najpopolnejšo zbirko (preko 300) pa v [GaJo79]. V reviji Journal of Algorithms v posebni rubriki že leta zbirajo nove in nove probleme z znano ali pa še neznano časovno zahtevnostjo.
Morda najbolj popularen problem kombinatorične optimizacije (oziroma operacijskih raziskav) je problem trgovskega potnika (angl. traveling salesman problem, TSP).



Slika 1: Korak lokalne optimizacije tipa 2–opt

TSP spada v širši razred transportnih problemov, ki so deležni precejšnje pozornosti v strokovni literaturi. Za ilustracijo povejmo, da pregledni članek iz leta 1983 [BGAB83] navaja kar 699 referenc v zvezi s transportnimi problemi!
Ob mnogih primerih praktične uporabe je verjetno eden od razlogov popularnosti problema trgovskega potnika dejstvo, da ga je zelo enostavno opisati. Imamo neusmerjen graf z utežmi na povezavah (*omrežje*). *Hamiltonov obhod* ali *Hamiltonov cikel* je pot, katere začetna in končna točka sovpadata in ki obišče vsako točko grafa natanko enkrat. Problem je poiskati Hamiltonov obhod z minimalno dolžino, če za dolžino poti vzamemo vsoto uteži na povezavah poti.
V duhu definicije 1 so dopustne rešitve vsi Hamiltonovi obhodi na grafu. Vsak Hamiltonov obhod lahko zapišemo tudi kot ciklično permutacijo. če je graf poln, potem tudi vsaka ciklična permutacija definira Hamiltonov obhod, v splošnem pa to ni res. Stroškovna funkcija je

$$c(\pi) = \sum_{i=1}^{n} d(i, \pi(i)) \qquad (3)$$

kjer je π permutacija $\in S_n$, $\{1, \ldots, n\}$ so oznake točk, $d(i,j)$ pa je utež na povezavi (i,j).
Pogosto je uporabljana lokalna optimizacija tipa r–opt. Sosednje stanje k danemu stanju dobimo tako, da vzamemo r povezav in jih zamenjamo z novimi. Primer za 2–opt je na sliki 1.
Ker z rastočim r časovna zahtevnost lokalne optimizacije r–opt narašča, so najpogosteje uporabljane 2–opt in 3–opt, ali pa optimizacija, ki se sproti odloča za r. Kot najboljšo pa v [LLRS85] priporočajo Or–opt.
Problem trgovskega potnika je NP–poln. Tu samo omenimo, da so tudi mnogi zanimivi posebni primeri še vedno NP–polni problemi, na primer: simetrični TSP, Evklidski TSP, TSP na polnem omrežju, TSP za katerega velja trikotniška neenakost. Še več, že odločitveni problem, ali ima dani graf Hamiltonov cikel, je NP–poln problem.
Pogosto rešujemo problem na polnem omrežju, saj s tem ne izgubimo splošne uporabnosti algoritma. Vedno namreč lahko povezave, ki jih nočemo uporabljati, obtežimo z zelo veliko utežjo (na primer večjo kot vsota vseh dovoljenih uteži). Če je potem dobljena optimalna rešitev prevelika, vemo, da je bila uporabljena kakšna prepovedana povezava.
V primeru, ko rešujemo problem na splošnem grafu, bi bilo naravno vzeti za dopustne rešitve Hamiltonove obhode grafa. V tem primeru bi bil že algoritem $\mathcal{R}$ časovno zelo zahteven, saj bi moral reševati NP–poln problem. To je pomemben razlog za odločitev, da rešujemo problem na polnem omrežju.

4. Algoritem $\mathcal{SA}$

Eden od razlogov za popularnost algoritma $\mathcal{SA}$ je analogija z modelom procesa ohlajanja snovi iz statistične mehanike. Tu analogije ne bomo opisovali, omenimo le referenco [KiGV83].
S stališča kombinatorične optimizacije je algoritem ohlajanje ($\mathcal{SA}$) zelo enostaven. Za razliko od lokalne optimizacije so možni tudi premiki v smeri poslabšanja stroškovne funkcije. Verjetnost takega koraka je odvisna od parametra T, ki mu po analogiji s fizikalnim modelom rečemo temperatura.

Algoritem Ohlajanje ($\mathcal{SA}$):
generiraj slučajno začetno stanje
nastavi začetno temperaturo T
repeat
  znižaj temperaturo T
  slučajno izberi soseda
  if sosed je boljši **then** premakni se
  else (* sosed je slabši*)
    premakni se z verjetnostjo $p = p(T)$
until preveč korakov

Označimo s p verjetnost dogodka, da je v algoritmu sprejet korak, ki poslabša vrednost stroškovne funkcije. Ta verjetnost je odvisna od trenutnega stanja (trenutne dopustne rešitve) in od trenutne vrednosti parametra T. Predpostavili bomo, da je p padajoča funkcija parametra T. Pri danem primerku problema in dani temperaturi T bomo zahtevali, da naj bo p omejena s konstanto $p < P < 1$.
Funkcijska odvisnost je običajno oblike

$$p = \begin{cases} 1 & \Delta C < 0 \\ \exp^{-\Delta C/T} & \Delta C \geq 0 \end{cases} \qquad (4)$$

kjer smo z ΔC označili razliko stroškovnih funkcij starega in novega stanja. Taka definicija je predlagana v [MRRT53] in [KiGV83], in je običajno uporabljena. Edina avtorju znana izjema je definicija Boltzmanovega stroja [AaKo87, AaKo88] kjer uporabljajo

$$p = \left(1 + \exp^{-\Delta C/T}\right)^{-1} \qquad (5)$$

Definicija algoritma $\mathcal{SA}$ je res enostavna, vendar pri implementaciji hitro naletimo na netrivialne probleme. Na primer, kako je treba zmanjševati temperaturo, da bomo dosegli dobre rezultate. Vemo, da prehitro zniževanje temperature ne zagotavlja več 'konvergence' algoritma [MiRS86]. Ali znamo zniževanje temperature dobro določiti vsaj za kakšen poseben problem? V [LaAr87, Laar88] in v [LaDe88] predlagajo polinomske strategije ohlajanja, ki se dobro odrežejo na nekaterih preiskušenih primerih. (Pri tem se seveda morajo zadovoljiti z dobrimi namesto optimalnih rešitev.) Tu se s tem problemom ne bomo podrobneje ukvarjali.
Zniževanje temperature definiramo lahko tako, da povemo, kako je T odvisna od števila korakov. Poseben primer (in ekvivalenten !) je, da povemo, koliko korakov naredimo pri neki temperaturi. Odvisnost temperature od časa v tem primeru podamo z zaporedjem parov (temperatura, število korakov), na primer

$$(T_1, m_1), (T_2, m_2), \ldots, (T_k, m_k), \ldots$$

kar pomeni: naredi m_1 korakov pri temperaturi T_1,... naredi m_k korakov pri temperaturi T_k,...itd. Kot že omenjeno, mora temperatura padati počasi, če želimo, da algoritem z verjetnostjo 1 najde optimalno rešitev (v končnem, toda ne omejenem številu korakov) [MiRS86].
V nadaljevanju bomo predpostavili, da temperatura T konvergira proti 0 z naraščajočim številom korakov.

$$\lim_{n \to \infty} T(n) = 0 \qquad (6)$$

5. Računski rezultati z algoritmom $\mathcal{SA}$ (pregled literature)

V tem razdelku bomo s kratkimi opombami pospremili poročila o testiranju algoritma $\mathcal{SA}$ na različnih problemih. Pregled še zdaleč ni popoln, zajema pa (verjetno) večino 'klasičnih' člankov in nekaj poročil, ki jih je avtor bolj ali manj slučajno dobil v roke.
V že večkrat omenjenem članku Kirckpatrick s sodelavci [KiGV83] preizkusi algoritem $\mathcal{SA}$ na problemu trgovskega potnika (TSP) in na dveh problemih, ki se pojavita pri načrtovanju integriranih vezij (Cell Placement, Wiring). Članek je populariziral algoritem. Uspešna uporaba tu in še v celi vrsti poročil pomeni, da so bile dosežene dobre ali celo dotlej najboljše znane rešitve za kakšen primerek kakšnega problema. O času, potrebnem za računanje ne poročajo ali pa priznavajo, da je algoritem zelo potraten.
Nasploh običajno (z izjemami, ki jih navajamo v pregledu) velja splošna ugotovitev (ki jo v [AnMS87] (stran 8) povzamejo po [JAMS85] in [GoSk86]), da je $\mathcal{SA}$ tipično slabši od algoritmov, ki so posebej prilagojeni konkretnemu problemu.
Bonomi in Lutton [BoLu84] z algoritmom računata 'skoraj optimalne' rešitve za TSP v $O(n^2)$ časa. Eksperimentalno ugotovita, da se algoritem $\mathcal{SA}$ obnese bolje od lokalne optimizacije. Na žalost uporabita staro verzijo 2-opt algoritma. Ista avtorja v [LuBo86] algoritem uporabita na problemu minimalno uteženo Evklidsko prirejanje (angl. minimum weighted perfect Euclidean matching). Opazita, da je potrebni čas (za osnovno verzijo algoritma $\mathcal{SA}$) 'ogromen' (stran 195, zadnji odstavek). Še eno poročilo omenjenih avtorjev [BoLu86] se ukvarja z aplikacijo algoritma na kvadratni problem prirejanja (angl. Quadratic sum assignement problems). Algoritem $\mathcal{SA}$ je predlagan kot linearni aproksimacijski algoritem.
O poizkusih na kvadratnem problemu prirejanja (Quadratic assignement problem) poročata Burkhardt in Rendl [BuRe84]. Iz rezultatov je videti, da $\mathcal{SA}$ ni bil bistveno boljši od QAPH4B algoritma (ki je v resnici lokalna optimizacija), posebej če upoštevamo porabljeni čas. Uporabljena je tudi različica z več ponovitvami.
Grover [Grov87] poroča o uporabnem programu, ki prinaša lepe prihranke (precej odstotkov) na problemu razporejanja čipov (angl. Standard Cell Placement), uporabljanem v načrtovanju integriranih vezij (VLSI design). Začne vedno z dobro začetno rešitvijo.
Poročili [DoSS87, DoSk88] ugotavljata, da je algoritem $\mathcal{SA}$ na problemu razbitja grafa (graph partitioning) zelo potraten, so pa dobili dobre rešitve v primerjavi z nekaterimi drugimi algoritmi.
V [AnMS87] (stran 8) sta omenjeni dve eksperimentalni študiji. V [JAMS85] ugotavljajo, da je z $\mathcal{SA}$ sicer mogoče dobiti dobre rešitve, vendar je bil algoritem na problemih razcep števil (number partitioning), barvanje grafa (graph colouring) in trgovski potnik (TSP) praviloma slabši od tradicionalnih algoritmov za te probleme. V primeru

problema razbitja grafa so bili dobljeni rezultati ugodni za algoritem $\mathcal{SA}$. V drugem poročilu [GoSk86] ugotavljajo, da za problem trgovskega potnika in "p-mediane" obstajajo algoritmi, ki so boljši od $\mathcal{SA}$.

V članku [LALW89] Laarhoven, Aarts, Lint in Wille poročajo o doslej najboljših rešitvah za problem trdnjav (angl. Footbal-pool problem) na 6, 7 in 8 parih. Prej je najboljšo znano rešitev za $n = 6$ našel Wille [Will87], prav tako z algoritmom $\mathcal{SA}$.

Za konec omenimo še nekaj poročil, v katerih primerjajo algoritem $\mathcal{SA}$ z lokalno optimizacijo. V [LaAr87] povzemajo primer [LuMe86], za katerega je dokazano, da je $\mathcal{SA}$ boljši od $\mathcal{RLS}$ (pričakovano število korakov $\mathcal{SA}$ je manjše od pričakovanega števila korakov algoritma $\mathcal{RLS}$). Ta rezultat ne nasprotuje našemu izreku 2, saj je primer narejen za verzijo $\mathcal{SA}$ s konstantno temperaturo, torej ne ustreza pogojem izreka (oziroma naši definiciji algoritma $\mathcal{SA}$).

Rahin [RaSh89] poroča o primerjavi lokalne optimizacije in $\mathcal{SA}$ na problemu razbitja grafa. Za majhne n se je lokalna optimizacija izkazala za boljšo: rezultat je bil boljši in čas krajši. Uporabljen je bil stari Lin-Kerninghamov algoritem za lokalno optimizacijo.

Lam in Delosme [LaDe88] poročata o testiranju algoritma $\mathcal{SA}$ z originanim načinom ohlajanja. V nekaterih primerih se je algoritem izkazal za boljšega od Lin-Kerninghamove in od Karpove hevristike za problem trgovskega potnika. Na problemu razbitja grafa je bil pri nekaterih načinih generiranja slučajnih primerkov problema algoritem boljši, pri drugih pa slabši od hevristike Fiduccia in Mattheysesa.

Kategorično v prid $\mathcal{SA}$ interpretirajo rezultate v [JAMS85], kjer pa je to storjeno izrazito 'nepošteno'. 100 poskusov z algoritmom $\mathcal{SA}$ je primerjano s 100 poskusi lokalne optimizacije ($\mathcal{RLS}$), to pa pomeni, da je bil čas porabljen za $\mathcal{SA}$ kar približno 300-krat daljši! (6 minut za $\mathcal{SA}$ proti 1 sekundi za $\mathcal{RLS}$).

Po drugi strani v zborniku [AnMS87](stran 95) omenjajo referenco [RiTi85] kjer so eksperimentalno ugotovili, da je pri globalni optimizaciji na kompaktni množici $\mathbf{S} \subset \mathbb{R}^k$ pri pogoju, da leži optimalna točka v notranjosti $\mathbf{S}$ med znanimi najboljša naslednja metoda:

1.generiraj nekaj slučajno izbranih točk (z enakomerno porazdelitvijo);
2.naredi lokalno iskanje na vsaki od njih
3.zapomni si dotedaj najboljšo rešitev

(Torej analogno algoritmu $\mathcal{RLS}$ za diskretni primer!)

Pred zaključkom pregleda povzemimo komentar iz [LaAr87]. Večina poskusov z algoritmom $\mathcal{SA}$ je bila narejena z enostavnim ohlajanjem. Rezultati bi bili verjetno boljši, če bi uporabili bolj "zvite" načine ohlajanja, kot jih (na primer) predlagajo v [LaAr87, Laar88]. Po drugi strani je res, da tudi algoritmi, s katerimi $\mathcal{SA}$ primerjajo niso vedno najboljši od znanih. Pri tovrstnih poskusih je na žalost vedno tako, da je zelo težko doseči popolnoma pošteno primerjavo. Verjetno bo za dokončno splošno priznano sodbo potrebno še precej primerjalnih študij.

Poskusimo navezati nekatere rezultate poskusov na teoretični rezultat iz naslednjega razdelka: Upanje, da je mogoče rezultat izreka 2 podkrepiti še s kakšnim praktičnim rezultatom, sta naslednji poročili. Dueck in Scheuer [DuSc88] sta testirala algoritem $\mathcal{TA}$ (treeshold accepting), ki je nekoliko popravljena lokalna optimizacija. Namesto verjetnostnega kriterija ta algoritem sprejema poleg korakov navzdol tudi korake, ki 'ne gredo preveč navzgor'. Algoritem torej 'preskoči' majhne 'hribčke' in 'dolinice', ki so za običajno lokalno optimizacijo usodni. Doseženi rezultati na primerku problema trgovskega potnika s 442-mesti so najboljši doslej (boljši kot tisti, dobljeni z algoritmom $\mathcal{SA}$). Poročajo tudi o sorazmerno kratkem času računanja.

K temu dodajmo še primerjavo [LeBi89] med algoritmom $\mathcal{SA}$ in 'quenching' algoritmom (zelo hitro ohlajanje), ki se je prav tako izkazalo za boljše od počasnega ohlajanja (pri ustreznem številu ponovitev za hitrejši algoritem, seveda). Če poskusimo iz tega potegniti nekakšen nasvet, bi lahko rekli takole: Običajno se ne izplača predolgo časa porabiti z 'zelo premetenim' algoritmom. Bolje je večkrat začeti prav od začetka, če 'dovolj dolgo' nimamo zaželjenega uspeha. Obravnava vprašanja, kaj v konkretnih primerih pomeni 'dovolj dolgo' presega okvire tega dela. Tukaj omenimo le referenci [BoRV87] in [Laar88], ki obravnavata ustavitvena pravila za nekatere verjetnostne hevristike.

6. Konvergenca algoritma $\mathcal{SA}$

Konvergenco algoritma $\mathcal{SA}$ obravnavajo v člankih [GeGe84, Gida85] in [HaSa89], v [GeMi87] pa so omenjeni tudi izreki Hajeka in Tsitsiklisa. Model izvajanja algoritma je (časovno) nehomogena Markovska veriga, kajti prehodne verjetnosti se s časom spreminjajo. V tem razdelku bomo povzeli po enem od omenjenih virov izrek o konvergenci algoritma $\mathcal{SA}$.

Izrek 1 *[MiRS86] Če parameter T pada dovolj počasi, potem iz kakršnegakoli začetnega stanja algoritem $\mathcal{SA}$ z verjetnostjo 1 konča v enem od globalnih optimumov, če mu dovolimo narediti neskončno mnogo korakov.*

Ali, ekvivalentno: Če parameter T pada dovolj počasi, potem iz kakršnegakoli začetnega stanja algoritem $\mathcal{SA}$ z verjetnostjo poljubno blizu 1 konča v enem od globalnih optimumov, če mu dovolimo narediti dovolj korakov.

7. Primerjava algoritmov $\mathcal{SA}$ in $\mathcal{RLS}$

V tem razdelku navajamo izrek, ki pravi, da je algoritem $\mathcal{SA}$ asimptotsko slabši od algoritma $\mathcal{RLS}$. Še več, iz dokaza bo sledilo, da je celo enostavno ponavljanje neodvisnih generiranj dopustnih rešitev asimptotsko uspešnejši algoritem kakor algoritem $\mathcal{SA}$ [Žero88].

Uporabljali bomo naslednje oznake:

- N je število stanj (moč množice dopustnih rešitev)
- K_1 označuje število stanj, iz katerih vse poti, ki gredo samo navzdol, končajo v enem od globalnih optimumov.
- K označuje število stanj, iz katerih obstaja pot, ki gre samo navzdol in konča v enem od globalnih optimumov. Očitno $K \geq K_1 > 0$. Da se izognemo trivialnim primerom privzemimo še $N > K$.
- R je dolžina najdaljše poti, ki gre samo navzdol.
- w je razmerje med časom, potrebnim za generiranje začetne dopustne rešitve (algoritem $\mathcal{R}$) in med časom, potrebnim za generiranje novega stanja, če kakšno stanje že poznamo (algoritem $\mathcal{N}$).

- d je maksimalna stopnja stanja (maksimalno število sosedov) oziroma

$$d = \max_{x \in \mathbf{S}} |\mathrm{N}(x)|$$

- $1-p_i$ je verjetnost, da algoritem ne bo sprejel slabšega stanja v algoritmu $\mathcal{SA}$ pri temperaturi T_i. Lahko bi tudi rekli: verjetnost, da algoritem ne bo šel 'gor'. (Vrednost p_i je odvisna tudi od trenutnega stanja.)
- $q_i = \min\{1 - p_i\}$, kjer minimum izračunamo po vseh možnih stanjih danega primerka problema. Temperatura je tu fiksirana (T_i). q_i je torej spodnja meja za verjetnost dogodka, da pri temperaturi T_i pri danem primerku problema algoritem $\mathcal{SA}$ ne bo sprejel koraka 'gor', v smeri poslabšanja stroškovne funkcije. V dokazu Izreka 2 bomo predpostavili $q_i \to 1$ ko gre $i \to \infty$. Če vzamemo 4 za definicijo p in predpostavimo $T \to 0$ sledi $q_i \to 1$, torej je za algoritem $\mathcal{SA}$ predpostavka izreka izpolnjena.
- n je število korakov, ki jih je algoritem že opravil.

V dokazu privzamemo, da algoritem $\mathcal{R}$ generira vse dopustne rešitve z enako verjetnostjo. (Privzetek deloma poenostavi dokaz, vendar ni bistven za resničnost izrekov.) Dokaz je (presenetljivo) enostaven. Poiščemo spodnjo mejo za verjetnost, da algoritem lokalna optimizacija najde globalno optimalno rešitev v n korakih in zgornjo mejo za verjetnost, da algoritem $\mathcal{SA}$ najde globalno optimalno rešitev v n korakih. Potem primerjamo obe oceni in sledi rezultat (za dokaz glej [Žero88, Žero90]).
Formalno definiramo

$$P_{\mathcal{RLS}}(n) := Pr(\textit{algoritem } \mathcal{RLS} \textit{ je uspel v } n \textit{ korakih}) \tag{7}$$

in

$$P_{\mathcal{SA}}(n) := Pr(\textit{algoritem } \mathcal{SA} \textit{ je uspel v } n \textit{ korakih}) \tag{8}$$

ocenimo

Lema 1

$$P_{\mathcal{RLS}}(n) \geq 1 - \left(\frac{N - K_1}{N}\right)^{\lceil \frac{n}{Rd+w} \rceil} \tag{9}$$

Lema 2

$$P_{\mathcal{SA}}(n) \leq 1 - q_i^{\tilde{m}_i} q_{i-1}^{m_{i-1}} \ldots q_2^{m_2} q_1^{m_1} \left(1 - \frac{K}{N}\right) \tag{10}$$

kjer je $n = \tilde{m}_i + \sum_{j=1}^{i-1} m_j$ *in* $0 < \tilde{m}_i \leq m_i$.

in izpeljemo

Izrek 2 *Obstaja konstanta* n_0, *tako da velja*

$$\forall n > n_0 \Longrightarrow P_{\mathcal{SA}}(n) < P_{\mathcal{RLS}}(n) \tag{11}$$

8. Skoraj optimalne rešitve

Pri reševanju NP–težkih problemov se moramo običajno odpovedati zahtevi po optimalni rešitvi. Izkaže se, da je mogoče enako primerjavo med algoritmoma $\mathcal{SA}$ in $\mathcal{RLS}$ narediti tudi v primeru ko smo zadovoljni tudi s skoraj-optimalnimi rešitvami optimizacijskega problema, ki jih definiramo takole:

Bodi c_{min} strošek optimalne rešitve in bodi c_{max} strošek ene od najslabših rešitev. (torej $c_{min} \leq c \leq c_{max}$ za vsako dopustno rešitev) Izberimo si $\epsilon > 0$. Rešitev s stroškom $c \leq c_{min} + \epsilon(c_{max} - c_{min})$ imenujemo *ϵ-skoraj optimalna* rešitev.
Po definiciji ϵ govori o kvaliteti rešitve na nekako normirani stroškovni funkciji. Na primer za $\epsilon = 0.10$ skoraj optimalne rešitve niso za več kot 10% slabše od optimalne rešitve. Pri $\epsilon = 0$ dobimo optimalne rešitve, pri $\epsilon = 1$ pa kar vse dopustne rešitve.
Tukaj velja opozoriti, da so mnenja, katere rešitve je smiselno proglasiti za skoraj optimalne, deljena. Namesto o skoraj optimalnih rešitvah bomo tu raje govorili o množici *zaželjenih* rešitev, ki je lahko množica skoraj optimalnih rešitev za neki ϵ, lahko pa za množico zaželjenih rešitev vzamemo tudi kakšno drugo podmnožico množice S.
Če povzamemo oznake iz dokaza izreka 2, ponovno pa definiramo K_1, K in pojem 'uspeh algoritma $\mathcal{SA}$', potem lahko dokažemo podoben izrek za verjetnost zadetka nove množice zaželjenih rešitev.
V tem razdelku naj:

- K_1 označuje število stanj, iz katerih vse poti, ki gredo samo navzdol, končajo v eni od zaželjenih rešitev.
- K označuje število stanj, iz katerih obstaja pot, ki gre samo navzdol in konča v eni od zaželjenih rešitev. Očitno $K \geq K_1$.

Tokrat bomo rekli, da je algoritem $\mathcal{SA}$ uspel (najti zaželjeno rešitev) brž ko je dosegel stanje, iz katerega obstaja vsaj ena pot, ki gre samo navzdol in konča v stanju, ki ustreza zaželjeni rešitvi.
Z natanko istim premislekom kot v dokazu izreka 2 bi pokazali, da velja

Trditev 1 *Obstaja konstanta* n_0, *tako da velja*

$$\forall n > n_0 \Longrightarrow \tilde{P}_{\mathcal{SA}}(n) < \tilde{P}_{\mathcal{RLS}}(n) \tag{12}$$

kjer je $\tilde{P}_{\mathcal{A}}(n)$ *verjetnost, da je algoritem* $\mathcal{A}$ *našel zaželjeno rešitev v* n *korakih.*

Trditev lahko zapišemo še nekoliko drugače:

Posledica 1 *Bodi* $\tilde{P}_{\mathcal{SA}}(n_0) < p$ *in* $n_0 > \sum_{i=1}^{k-1} m_i$, *kjer je*

$$k = \min\left\{i; \forall j \geq i : q_j > \left(1 - \frac{K_1}{N}\right)^{\frac{1}{Rd+w}}\right\}$$

Označimo $n_{\mathcal{RLS}} = \min\{n; \tilde{P}_{\mathcal{RLS}}(n) > p\}$ *in* $n_{\mathcal{SA}} = \min\{n; \tilde{P}_{\mathcal{SA}}(n) > p\}$. *Potem je*

$$n_{\mathcal{SA}} > n_{\mathcal{RLS}}$$

Torej je pri dani zahtevani kvaliteti rešitve in pri dani (dovolj veliki) minimalni zahtevani verjetnosti uspeha število potrebnih korakov manjše, če se odločimo za algoritem $\mathcal{RLS}$ namesto za algoritem $\mathcal{SA}$.

9. Vzporedne implementacije algoritma $\mathcal{SA}$

V tem razdelku predstavimo izrek, analogen izreku 2, ki velja za vse (nam) poznane paralelizacije algoritma $\mathcal{SA}$. V literaturi sta znana dva modela paralelizacije algoritma $\mathcal{SA}$. Prvi [BoLu84, FęKO85] temelji na delitvi podatkov,

kar v primeru problema trgovskega potnika pomeni, da cikel, ki predstavlja začetno rešitev, razbijemo na več poti in jih razdelimo procesorjem. Vsak od procesorjev nekaj časa optimizira svoj del poti (lahko si mislimo, da sta začetna in končna točka fiksirani), potem pa si procesorji izmenjajo podatke. S tem je doseženo, da lahko algoritem (s pozitivno verjetnostjo) doseže katerokoli dopustno rešitev. Drugi model vsem procesorjem dodeli nerazdeljeni problem, procesorji na njem vsak zase izvajajo običajni algoritem $\mathcal{SA}$, po določenem času pa se 'sinhronizirajo'. Sinhronizacija tu pomeni, da na osnovi dotlej najboljših rešitev izberejo nove začetne rešitve. Preizkušeni so bili različni kriteriji sinhronizacije, na primer izbira najboljše rešitve, izbira n najboljših rešitev, slučajna izbira, na primer glede na Boltzmanovo porazdelitev, itd. [LaAr87, DoSS87, ABHL86].

V nadaljevanju bomo definirali model, ki zajema vse zgoraj naštete različice kot posebne primere. Za ta splošni model se da pokazati analogni izrek Izreku 2. Ker je mogoče algoritem lokalna optimizacija $\mathcal{RLS}$ naravno implementirati na več procesorjih z optimalnim pospeškom (saj nimamo nobenih izgub zaradi komunikacije oziroma sinhronizacije), ni presenetljivo, da je tudi na več procesorjih $\mathcal{SA}$ asimptotsko slabši od lokalne optimizacije. (Pospešek pri paralelnem izvajanju je definiran s takoimenovanim Gustafsonovim zakonom [Gust88], ki je na glavo postavljena oblika Amdahlovega zakona [Amda67]. Gustafsonov zakon mnogo očitneje pokaže moč vzporednega izvajanja algoritmov, zato je bil seveda lepo sprejet v krogih, ki se s tem ukvarjajo. Na kratko in neformalno povedano zakona povesta naslednje: če imamo na voljo n procesorjev optimalni pospešek pomeni, da bomo porabili n-krat manj časa kot prej. Ali, še lepše, na n procesorjih ob optimalnem pospešku lahko v istem času rešimo n-krat večje probleme.)

V splošnem modelu predpostavljamo, da p procesorjev (neodvisno) izvaja splošni algoritem, imenovali ga bomo $\mathcal{PSA}$, s sinhronizacijsko razporeditvijo (angl. sinchronization schedule). Namesto 'temperature' (ki je v splošnem lahko različna od procesa do procesa) tu uporabimo števec korakov algoritma in verjetnost, da bo algoritem sprejel korak 'gor', je zdaj funkcija števila korakov $p = p(i)$. Zaporedje $0, v_1, v_2, \ldots, v_k, \ldots$ zdaj beremo: naredi v_1 korakov do prve sinhronizacije, potem naredi v_2 do druge sinhronizacije, itd. Oziroma, sinhroniziraj se prvič pri v_1-tem koraku, drugič pri $v_1 + v_2$-tem koraku, itd. Algoritem zapišemo podobno kot prej:

Algoritem $\mathcal{PSA}$:
generiraj slučajno začetno stanje
$i := 0$;
repeat
  repeat
    i:=i+1;
    slučajno izberi soseda
    if sosed je boljši **then** premakni se
    else (* sosed je slabši *)
      premakni se z verjetnostjo $p = p(i)$
  until sinhronizacija
  izberi novo začetno stanje in/ali
  izmenjaj podatke
until preveč korakov

Oba prej omenjena pristopa k paralelizaciji $\mathcal{SA}$ sta posebna primera splošnega algoritma PSA.

Verjetnost koraka 'gor' zdaj ni odvisna od parametra T, ampak od i. Ponovno bomo uporabili iste oznake kot v dokazu izreka 2. Nekoliko drugače bomo definirali samo verjetnost

$$q_i = \min\{ \textit{ verjetnost, da } \mathcal{PSA} \textit{ ne gre 'gor' v fazi } i \textit{ na nobenem od procesorjev}\} \quad (13)$$

kjer minimum izračunamo po vseh možnih stanjih danega primerka problema. Izraz "faza i" tu nadomešča frazo "med sinhronizacijama $i-1$ in i". V dokazu Izreka 3 bomo ponovno predpostavili $q_i \to 1$ ko gre $i \to \infty$. To bo gotovo res, če se bo sistem 'ohlajal', ali, z drugimi besedami, če bo maksimalna temperatura (po vseh procesorjih) kakorkoli šla proti 0.

Verjetnost, da p neodvisnih izvajanj algoritma lokalna optimizacija najde globalni optimum v n vzporednih korakih je očitno omejena z istim tipom spodnje meje kot če imamo samo eno izvajanje.

$$P_{\mathcal{PRLS}}(n) = 1-(1-P_{\mathcal{RLS}}(n))^p \geq 1-(1-(1-Q))^p = 1-Q^p \quad (14)$$

kjer je

$$Q = \left(\frac{N-K_1}{N}\right)^{\lceil \frac{n}{Rd+\omega} \rceil} \quad (15)$$

Analogno kot prej bomo rekli, da je algoritem $\mathcal{PSA}$ uspel, brž ko je vsaj eden od procesorjev našel stanje, iz katerega pelje vsaj ena pot, ki gre samo navzdol, in konča v enem od globalnih optimumov.

Podobno kot prej dobimo

Lema 3 *Za* $n = \sum_{k=1}^{i-1} v_k + \tilde{v}_i$, *kjer je* $0 < \tilde{v}_i \leq v_i$, *velja*

$$P_{\mathcal{PSA}}(n) \leq 1 - q_i^{p\tilde{v}_i} q_{i-1}^{v_{i-1}} \ldots q_2^{v_2} q_1^{v_1} \left(1 - \frac{K}{N}\right) \quad (16)$$

In

Izrek 3 *Obstaja konstanta* n_0, *tako da velja*

$$\forall n > n_0 \Longrightarrow P_{\mathcal{PSA}}(n) < P_{\mathcal{PRLS}}(n) \quad (17)$$

Podrobnosti opuščamo. (Vsebina razdelka in dokaz izreka je rezultat krajšega obiska v laboratoriju za paralelne algoritme na Ecole Normale Superiuere de Lyon in je objavljena v [BFZ89a, BrFZ89, Žero90].)

Opomba: V posebnem primeru, ko vzporedno poganjamo p neodvisnih procesov, lahko zgornjo mejo za algoritem $\mathcal{PSA}$ nekoliko izboljšamo. V tem primeru imamo namreč p neodvisnih poskusov, od katerih je vsak en običajni (zaporedni) tek algoritma $\mathcal{SA}$.

$$P^*_{\mathcal{PSA}} \leq 1 - \left(q_i^{\tilde{m}_i} q_{i-1}^{m_{i-1}} \ldots q_2^{m_2} q_1^{m_1} \left(1 - \frac{K}{N}\right)\right)^p \quad (18)$$

Razliko pojasnimo preprosto: V rezultatu leme, enačba (16), 'manjkajoča' potenca p na faktorju $(1 - \frac{K}{N})$ je posledica naše implicitne predpostavke, da smo vzporedni tek algoritma začeli z enim začetnim stanjem (in ne s p-timi).

Lahko bi se prepričali, da izrek 3 tudi v tem primeru velja.

10. Zaključek

Namen sestavka je bil predstavitev algoritma SA, ki je bil v zadnjih letih deležen precej pozornosti v strokovni javnosti. Pregled teoretičnih in praktičnih rezultatov kaže, da je vrednost algoritma v splošnem še nejasna. Asimptotski rezultat, ki ga predstavimo, implicira, da v vseh primerih algoritem ne more biti najboljši. Verjetno je relativna uspešnost algoritma (glede na tekmece) bistveno odvisna od lastnosti prostora stanj, ki je pri različnih problemih kombinatorične optimizacije lahko bistveno različen. Omenimo tukaj sveži članek [Sasa91], ki dokazuje povezavo med 'gostoto' prostora stanj in mejo za uspešnost Metropolisovega algoritma. Potrebno bo še precej eksperimetalnih študij, preden bo jasno, v katerih primerih se Ohlajanje res izplača.

Zahvala: Članek temelji na delu disertacije, ki je nastala pod mentorstvom prof. Tomaža Pisanskega. Zahvaljujem se mu za pomoč in vzpodbudo. Zadnja verzija članka je nastala med avtorjevim obiskom prof. Wilfrieda Imricha na Montanuniversität v Leobnu.

Literatura

[AaKo87] E.H.L.Aarts, J.H.M.Korst, Boltzmann Machines and Their Applications *PARLE Parallel Architectures and Languages Europe*, Lecture Notes in Comp. Sci. 258, Springer, Berlin 1987, 34-50

[AaKo88] E.H.L.Aarts, J.H.M.Korst, Computations in massively parallel networks based on the Boltzmann machine: A rewiew, *Parallel Computing*, 9 (1988/89) pp. 129-145

[ABHL86] E.H.L.Aarts, F.M.J.Bont, E.H.A.Habers, P.J.M. Laarhoven: Parallel Implementations of the Statistical Cooling Algorithm, *INTEGRATION, the VLSI journal* 4 (1986) pp. 209-238

[Amda67] G.M. Amdahl: Validity of the single-processor approach to achieving large scale computing capabilities. *AFIPS Conference Proceedings, Atlantic City, N.J., Apr.18-20* **30** AFIPS Press, Restn, Va. 1967 pp. 483-485

[AnMS87] G. Andreatta, F. Mason, P. Serafini (eds.), *Advanced School on Stochastics in Combinatorial Optimization*, World Scientific, Singapore 1987

[BGAB83] L. Bodin, B.Golden, A.Assad, M.Ball, Routing and Scheduling of Vechicles and Crews, the State of the Art *Comput & Ops. Res.* **10** (1983) pp. 63-211

[BoLu84] E.Bonomi, J.L.Lutton, The N-city traveling salesman problem: Statistical Mechanics and the Metropolis algorithm, *SIAM rewiev* **26** 1984

[BoLu86] E.Bonomi, J.L.Lutton, The asymptotic behaviour of quadratic sum assignement problems: A statistical mechanics approach, *European journal of Operational Research* **26** 1986 pp. 295-300

[BoRV87] G.E.Boender, A.H.G. Rinooy Kan, C. Vercelis: Stochastic Optimization Methods, v *Advanced School on Stochastics in Combinatorial Optimization*, G. Andreatta, F. Mason, P. Serafini (eds.):, World Scientific, Singapore 1987

[BrFZ89] B.Braschi, A.G.Ferreira, J. Žerovnik: On the Behaviour of Simulated Annealing, *Research Report no.89-10*, 1989, LIP-IMAG, Lyon, France

[BFZ89a] B.Braschi, A.G. Ferreira, J.Žerovnik: On the Asymptotic Behaviour of Parallel Simulated Annealing, v Parallel Computing 89 (eds: D.J.Evans, G.R. Joubert, F.J.Peters) North-Holland, Amsterdam 1990, 263-268

[BuRe84] R.E.Burkard, F.Rendl, A thermodinamically motivated simulation procedure for combinatorial optimization problems, *European Journal of Operational Research* **17** 1984 pp. 169-174

[Černe84] V. Černy, A thermodynamical approach to the travelling salesman problem: An efficient simulation algorithm, *Journal of Optimization Theory and Applications* **45** 1984 pp. 41-51

[DoSS87] J.G.Donnett, M.Starkey, D.B.Skilicorn, Effective Algorithms for Partitioning Distributed Programs, *(preprint) Queen's University*, Kingston, Canada 1987

[DoSk88] J.G.Donnett, D.B.Skilicorn, Code Partitioning by Simulated Annealing, *v: Parallel Processing and Applications, E.Chiricozzi and A.D'Amico (eds.)*, North-Holland, 1988

[DuSc88] G.Dueck and T.Scheuer, Threshold Accepting, A General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing, *(preprint) Heildelberg Scientific Center TR 88.10.011* 1989

[FeKO85] * E. Felten, S.Karlin and S.W.Otto: The traveling salesman problem on a hypercubic, MIMD computer *Proceedings of the 1985 International Conference on Parallel Processing* Avgust 20-23, 1985

[Frie79] R. Friedvals, Fast Probabilistic Algorithms, Lecture Notes in Computer Science 74, Springer-Verlag, Berlin, 1979, pp.57-69

[GaJo79] M.R. Garey, D.S. Johnson , *Computers and Intractability*, W.H. Freeman and Co., San Francisco (1979)

[GeGe84] * S.Geman, D.Geman, Stochastic Relaxation, Gibbs Distributions and the Bayesian Restoration of Images, IEEE Trans PAMI 6 (1984)

[GeMi87] S.B. Gelfand, S.K. Mitter, Simulated Annealing v *Advanced School on Stochastics in Combinatorial Optimization, (eds:) G. Andreatta, F. Mason, P. Serafini*, World Scientific, Singapore 1987

[Gida85] * B.Gidas, Nonstationary Markov Chains and Convergence of the Annealing Algorithm, *J.Stat. Phys.* **39** (1985) pp. 73-131

[GoSk86] B.Golden and C.Skiscim: Using Simulated Annealing to Solve Routing and Location Problems, *Naval Res. Log. Quarterly* **33** 1986 pp. 261-279

[Gust88] J.L. Gustafson: Reevaluating Amdahl's Law, *Communications of the ACM* **31** 1988 pp. 532-533

[Grov87] L.K.Grover, GRIM: A Fast Simulated Annealing Program for Standard Cell Placement *IEEE 1987 Custom Integrated Circuits Conference* pp. 622-624 1987

[HaSa89] H. Haario, E. Saksman, On the Definition and Convergence of the Simulated Annealing Process in General State Space, Reports of the Dept. of Math., University of Helsinki, 1989

[HoUl86] V.E.Hopcroft, J.D.Ullman: Uvod v teorijo avtomatov,
jezikov in izračunov, (prevod B.Vilfan), Fakulteta za elektrotehniko, Ljubljana 1986

[JAMS85] D.S.Johnson, C.R.Aragon, L.A.McGeoch and C.Schevon: Optimization by simulated Annealing: An Experimental Evaluation preprint, 1985

[John90] M.E. Johnson, *Simulated Annealing & Applications*, American Science Press, New York

[KiGV83] S.Kirckpatrick, C.D.Gellat, Jr., M.P.Vecchi, Optimization by Simulated Annealing *Science* 1983 **220** pp. 671-680

[KhSV81] * A.Khačaturjan, S.Semenovskaja and B.Vainštein: The Thermodynamical Approach to the Structure Analysis of Chrystals *Acta Cryst* **A37** 1981 pp. 742-754

[Kern86] W.Kern: A Probabilistic Analysis of the Switching Algorithm for the Euclidean TSP, Universität zu Köln, Report No 86.28.

[KLPP86] S. Klavžar, M.Lokar, M. Petkovšek, T. Pisanski, *Diskretna optimizacija* DMFA SRS, Ljubljana, 1986

[Knut81] D.E.Knuth: Algorithms in modern mathematics and computer science, Lecture Notes in Comp. Sci. 122, Springer-Verlag, Berlin 1981

[Koza86] J.Kozak, *Podatkovne strukture in algoritmi*, DMFA SRS, Ljubljana, 1986

[LALW89] Laarhoven, P.J.M., Aarts, E.H.L., Lint, J.H. and Wille, L.T.: New Upper Bounds for the Footbal Pool Problem for 6,7 and 8 Matches *Journal of Combinatorial Theory, Series A*, **52** pp. 304-312 (1989)

[LaAr87] P.J.M. Laarhoven, and E.H.L. Aarts, *Simulated Annealing, Theory and Applications*, D.Reidel Publishing Company, Dordrecht (1987)

[Laar88] P.J.M. Laarhoven: *Theoretical and computational aspects of simulated annealing*, Centrum voor Wiskunde en Informatica, Amsterdam 1989

[LaDe88] J. Lam, J-M. Delosme, An Efficient Simulated Annealing Schedule: Derivation, Report 8816 in An Efficient Simulated Annealing Schedule: Implementation and Evaluation, Report 8817, Yale University, 1988

[LLRS85] E.L. Lawler, J.K. Lenstra, A.H.G. Rinooy Kan, D.B. Schmoys (eds.), *The Traveling Salesman Problem*, John Wiley & sons (1985)

[LeBi89] C. Lee and L. Bic Comparing Quenching and Slow Simulated Annealing on the Mapping Problem *Proceedings of the third annual parallel processing symposium, California State University, Fullerton* 1989

[LITT89] D.C.Llewellyn, C.Tovey and M.Trick: Local optimization on graphs, *Discrete Applied Math.* **23** 1989 pp. 157-178

[LuMe86] M.Lundy, A.Mees, Convergence of an annealing algorithm, *Mathematical programming* **34** 1986 pp. 111-124

[LuBo86] J.L.Lutton, E.Bonomi, Simulated annealing algorithm for the minimum weighted perfect euclidean matching problem, *R.A.I.R.O. Oper. Res.* **20** 1986 pp. 177-197

[Maff86] F.Maffioli, Randomised algorithms in combinatorial optimisation: a survey, *Discrete Applied Mathematics* **14** 1986 pp. 157-170

[MRRT53] N.Metropolis, A.Rosenbluth, M.Rosenbluth, A.Teller, E.Teller, Equation of State Calculations by Fast Computing Machines, *J.Chem. Phys.* **21** 1953 pp. 1087-1091

[MiRS86] D.Mitra, F.Romeo, A. Sangiovanni-Vincentelli, Convergence and Finite Time Behavior of Simulated Annealing, *Adv. Appl. Prob.* **18** 1986 pp. 747-771

[Pinc70] * M.Pincus: A Monte Carlo Method for the Approximate Solutions of Certain Types of Constrained Optimization Problems *Oper. Res* **18** 1970 pp. 1225-1228

[Rabi76] M.O.Rabin: Probabilistic Algorithms, v Algorithms and Complexity (ur. Traub), 21-39, Academic Press 1976

[RaSh89] M.A.Rahin, J.Shield, Parallel Partitioning of Concurrent VLSI Simulation Graphs, *(preprint) Dept of Electronic and Electrical Engineering, Loughborough University of Technology*, 1989

[RiTi85] * A.H.G. Rinooy Kan, G.T. Timmer, Stochastic Global Optimization Methods, parts I & II, Econometric Institute, Erasmus University, Rotterdam, 1985

[SaHa88] G.H.Sasaki, B.Hajek, The time Complexity of Maximum Matching by Simulated Annealing, *Jour. ACM* **35** 1988 pp. 387-403

[Sasa91] G.Sasaki The effect of the density of states on the Metropolis algorithm *Information Processing Letters* **37** (1991) pp. 159-163

[Žero88] J. Žerovnik, Comparison of Asymptotic Behaviour of two Randomised Heuristic Approaches, *Preprint Series Dept. Math, University E.K. Ljubljana* **26**, pp. 186-192 (1988) (poslano v objavo)

[Žero90] J. Žerovnik, Verjetnost v kombinatorični optimizaciji, Univerza v Ljubljani, FNT, (oddana) doktorska disertacija, 1990

[Wels83] D.J.A.Welsh: Randomised algorithms, Discrete Applied Math. 5 (1983), 133-145

[Will87] L.T. Wille: The Footbal Pool Problem for 6 Matches: A new Upper Bound Obtained by Simulated Annealing, *Journal of Combinatorial Theory, Series A*, **45** pp. 171-177 (1987)

Z zvezdico * smo označili posredne reference.