Optymalizacja (minimalizacja) metodą Monte Carlo

Zagadnienie optymalizacji – sformułowanie problemu

- Problem optymalizacji wartości funkcji możemy sformułować jako zagadnienie poszukiwania wartości ekstremalnych, tj. maksimum bądź minimum. Maksimum możemy zamienić w minimum przemnażając funkcję przez czynnik (-1), zatem proces optymalizacji sprowadzany jest zazwyczaj do poszukiwania minimum wartości funkcji.

- minimum może być:
  - globalne (tego zazwyczaj szukamy)
  - lokalne (to zazwyczaj przeszkadza)

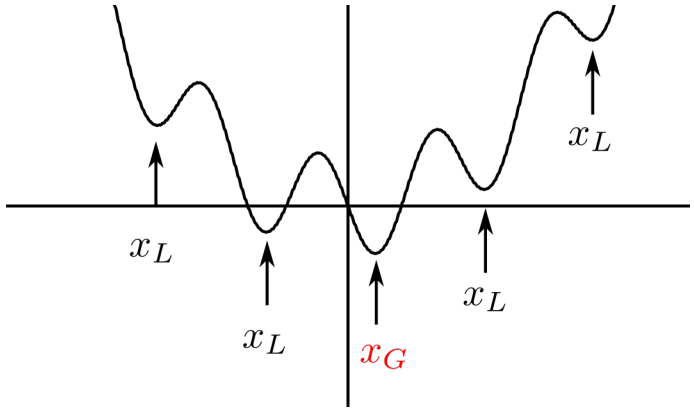
Zdefiniujmy matematycznie punkt w którym funkcja posiada minimum globalne (x_G)

$$f : R^n \to R$$

$$\min f(\vec{x}) = f(\vec{x}_G) \iff \bigwedge_{\vec{x} \in R^n} f(\vec{x}_G) < f(\vec{x})$$

oraz punkt stanowiący minimum lokalne (x_L)

$$\exists \varepsilon : \varepsilon > 0, \varepsilon \in R \quad \bigwedge_{\|\vec{x} - \vec{x}_L\| < \varepsilon} \quad f(\vec{x}_L) < f(\vec{x}) \quad \wedge \quad f(\vec{x}_L) > f(\vec{x}_G)$$



„Klasyczne” metody poszukiwania minimum wartości funkcji to metody iteracyjne

- (1D) metoda złotego podziału, metoda interpolacji kwadratowej Powella
- (>1D) metoda sprzężonego gradientu, największego spadku, Newtona (metody gradientowe)

  lub metoda Simplex (bezgradientowa)

- wadą metod standardowych jest to że potrafią znaleźć minimum zazwyczaj, gdy punkt startowy znajduje się w jego pobliżu (tak działają metody gradientowe)
- metody te znajdują pojedyncze minimum, więc nie ma pewności czy jest to minimum lokalne czy globalne (chociaż tej pewności na ogół nie mamy także w przypadku metody MC)
- zastosowanie standardowego podejścia sprawdza się w przypadku gdy liczba wymiarów przestrzeni, w której poszukujemy rozwiązania jest niewielka, rzędu 1-10
- klasyczne metody działają tylko dla problemów zdefiniowanych przy pomocy funkcji ciągłych, w przypadkach dyskretnych (np. problemy kombinatoryczne) stają się bezużyteczne

Przykład - problem komiwojażera

Dany jest zbiór miast które komiwojażer chce odwiedzić tylko raz i w takiej kolejności aby trasa je łącząca była najkrótsza i zamknięta.

położenia miast $\vec{d} = \{\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_n\}$

$$\min S = \sum_i \sum_{j>i} c_{ij} \cdot s_{ij} + s_{fl}, \qquad s_{ij} = |\vec{r}_i - \vec{r}_j|$$

s_{lf} – odcinek pomiędzy ostatnim a pierwszym miastem

$$c_{ij} = \begin{cases} 0, & \text{brak połaczenia} \\ 1, & \text{miasta połaczone} \end{cases}$$

Ponieważ ważna jest kolejność odwiedzanych miast więc moglibyśmy dokonać zmiany kolejności pierwotnej miast a następnie długość trasy liczyć jako odległości między kolejnymi parami – sąsiadującymi na liście miastami

pierwotne położenie miast

$$\vec{d} = \{\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_n\}$$

wektor indeksów miast zgodny z miejscem na liście

$$P_1 = \{1, 2, 3, \ldots, n\}$$

optymalna kolejność miast

$$\vec{d}_\alpha = \{\vec{r}_{\alpha_1}, \vec{r}_{\alpha_2}, \ldots, \vec{r}_{\alpha_n}\}$$

wektor indeksów dla optymalnej trasy

$$P_\alpha = \{\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n\}$$

$$\min S = \sum_{i=1}^{n-1} s_{\alpha_i \alpha_{i+1}} + s_{\alpha_n \alpha_1}$$

- S to **funkcja kosztu**, szukamy takiej kombinacji miast aby znaleźć jej minimum

Zauważmy że wektor P_α jest w zasadzie permutacją ciągu liczb zawartych w P_1.
Ile ścieżek możemy skonstruować?

$$N_\alpha = n!$$

$$n = 10 \rightarrow N_\alpha = 10! = 3.6 \cdot 10^6$$

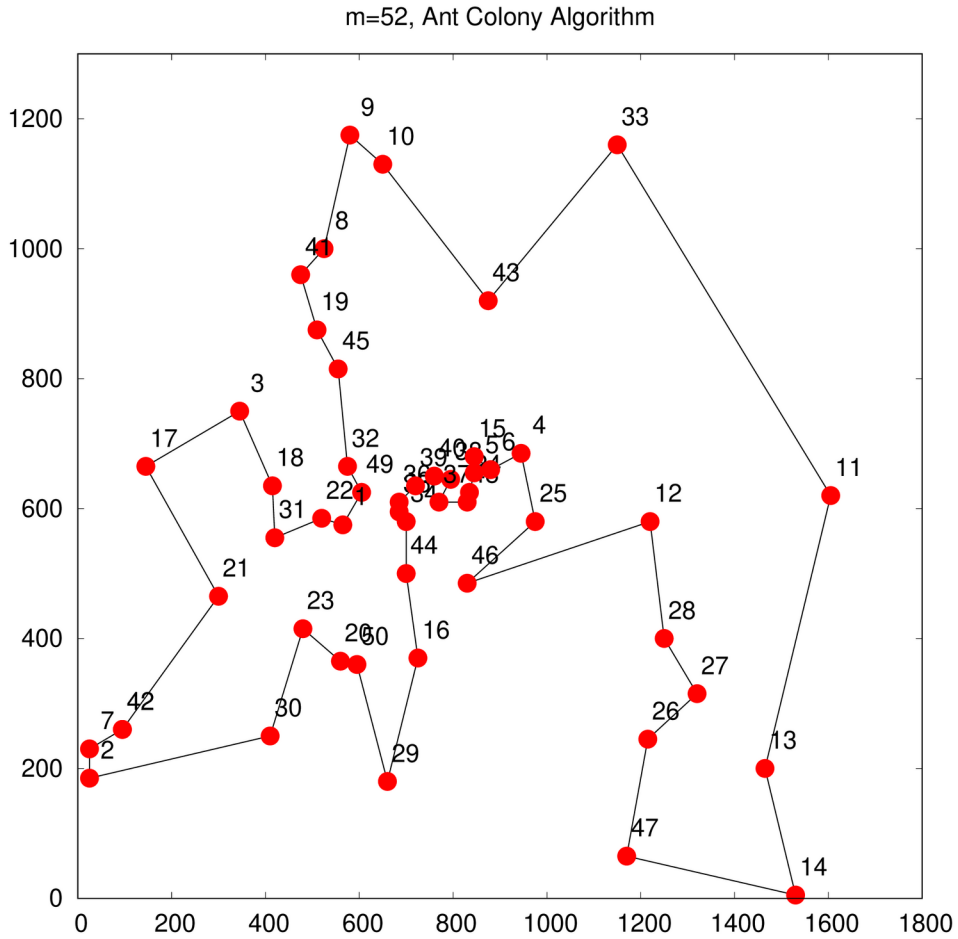
$$n = 20 \rightarrow N_\alpha = 20! = 2.4 \cdot 10^{18}$$

$$n = 50 \rightarrow N_\alpha = 50! = 3 \cdot 10^{64}$$

- metoda „brute force” polegająca na sprawdzaniu kolejnych permutacji jest w tym przypadku bezużyteczna
- jedynie metody stochastyczne (Monte Carlo) są w stanie poradzić sobie z tym problemem

Przykład

- 52 miasta
- najkrótszą trasę znaleziono algorytmem ewolucyjnym (ant colony algorithm)
- czas trwania symulacji: 2 sekundy



- indeksy przy punktach oznaczają położenie miasta na pierwotnej liście (wejściowej)

Przykład. Poszukiwanie konfiguracji układu oddziałujących ciał o najmniejszej energii np. klastra atomów, wycinka kryształu itp.

$$E_{tot} = V(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_n) = \underbrace{\sum_i V_1(\vec{r}_i)}_{\substack{\text{pola} \\ \text{zewnetrzne}}} + \underbrace{\sum_i \sum_{j>i} V_2(\vec{r}_i, \vec{r}_j)}_{\text{dwucialowe}} + \underbrace{\ldots}_{\substack{\text{trojcialowe} \\ \text{inne}}}$$

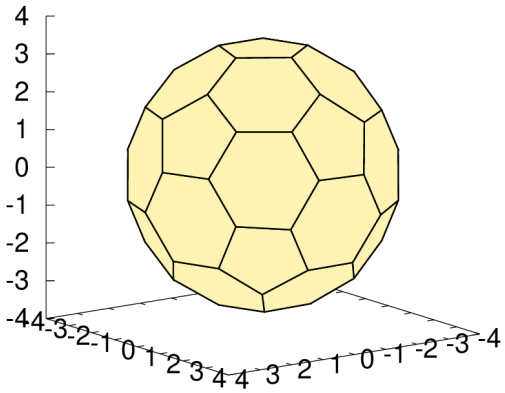
Znalezienie minimum energii staje się trudne z dwóch powodów:

- liczba cząstek w układzie określa wymiar przestrzeni, w której pracujemy (d to liczba współrzędnych opsiujących pojedynczą cząstkę)

$$N = d \cdot n, \quad d = 1, 2, 3$$

- w zależności od tego jak skomplikowaną postać ma potencjał oddziaływania zależy liczba lokalnych minimów funkcji kosztu (energii całkowitej), mogą one wręcz uniemożliwić znalezienie minimum globalnego

N=60, max 3-nearest neaighbours, E_{tot}= -421.2 eV



Rozkład Boltzmanna

W metodzie symulowanego wyżarzania wykorzystujemy dwa czynniki:

- rozkład Boltzmanna
- łańcuch Markowa

Rozkład Boltzmanna został zdefiniowany dla układów opisywanych przy użyciu **zbioru kanonicznego**, ustalone są wówczas parametry: N, V, T
(N -liczba cząstek, V – objętość układu, T – temperatura bezwzględna)

Określa on prawdopodobieństwo realizacji mikrostanu o energii E_i

$$p(E_i) = p_i = \frac{\exp\left(-\frac{E_i}{kT}\right)}{Z} = \frac{\exp\left(-\frac{E_i}{kT}\right)}{\sum_j \exp\left(-\frac{E_j}{kT}\right)}$$

Z jest funkcją rozdziału i pełni rolę czynnika normalizacyjnego.
Zazwyczaj jej nie znamy bo nie wiemy jakie są energie E_i.

Ponieważ zakładamy, że tylko N,V,T są stałe, więc energia układu może się zmieniać w czasie. To oznacza że dla T>0 układ może znajdować się w stanach o różnej energii, ale wyraźnie będą preferowane stany o niższej energii (większe p(E)) niż te o wysokiej.

Dysponując wyrażeniem określającym prawdopodobieństwo obsadzenia stanów, moglibysmy określić wartość oczekiwaną energii układu dla danej temperatury T

$$\langle E \rangle_{NVT} = \sum_j E_j p_j$$

Łańcuch Markowa, algorytm Metropolisa

Pomimo prostoty wyrażenia nie jesteśmy w stanie od razu wyznaczyć wartości oczekiwanej energii, gdyż nie znamy Z.
Zauważmy, że energia układu jest funkcją położeń i pędów cząstek – opisywanego wektorem w przestrzeni fazowej

$$E_i = E_i(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_n, \vec{p}_1, \vec{p}_2, \ldots, \vec{p}_n) = E_i(X)$$

zmiana jednej współrzędnej zmienia położenie w przestrzeni fazowej czyli stan X->Y

- różne realizacje wektora X będą odpowiadały różnym prawdopodobieństwom realizacji tych stanów
- prawdopodobieństwo realizacji stanu w przestrzeni fazowej (fgp) zapiszmy jako f(X)
  z warunkiem normalizacji

$$\int_\Omega f(X)dX = 1, \qquad dX = dr_1 dr_2 \ldots dp_1 dp_2 \ldots$$

- w warunkach równowagi termodynamicznej prawdopodobieństwa przejścia
  pomiędzy stanami X i Y oraz Y i X powinny być identyczne

$$\underbrace{K(X|Y)f(Y)}_{Y\to X} = \underbrace{K(Y|X)f(X)}_{X\to Y}$$

Częstość przejść K zdefiniujmy jako iloczyn częstości T (którą sami określimy)
oraz nieznanego prawdopodobieństwa akceptacji A

$$A(X|Y)T(X|Y)f(Y) = A(Y|X)T(Y|X)f(X)$$

$$A(X|Y) = A(Y|X)\frac{T(Y|X)f(X)}{T(X|Y)f(Y)} = A(Y|X)q(X|Y)$$

Nie znamy A(Y|X), więc chcąc znaleźć A(X|Y) musimy oprzeć się jedynie na znajomości wartości q(X|Y)

Metropolis zaproponował aby prawdopodobieństwo akceptacji nowego stanu określać na podstawie wyrażenia

$$p_{acc} = A(X|Y) = \min\{1, q(X|Y)\}$$

Od nas zależy postać wyrazu określającego cząstość przejść T.
Najprościej jest założyć

$$T(X|Y) = T(Y|X)$$

Ponadto wykorzystując fakt że fgp w przestrzeni fazowej ma postać

$$f(X) = \frac{\exp\left(-\frac{E(X)}{kT}\right)}{Z}$$

możemy w łatwy sposób określić prawdopodobieństwo akceptacji nowego stanu

$$q(X|Y) = \frac{T(X|Y)\exp\left(-\frac{E(X)}{kT}\right)}{T(X|Y)\exp\left(-\frac{E(X)}{kT}\right)} = \exp\left(-\frac{E(X) - E(Y)}{kT}\right)$$

$$\mathrm{p}_{acc} = \min\left\{1, \exp\left(-\frac{E(X)-E(Y)}{kT}\right)\right\}$$

- wzór Metropolisa

Prawdopodobieństwo akceptacji zaproponowane przez Metropolisa nie jest jedynym wyborem.
Alternatywnie można je określić następująco (**wzór Glaubera**)

$$p_{acc}^{G} = A'(X|Y) = \frac{q(X|Y)}{1+q(X|Y)}$$

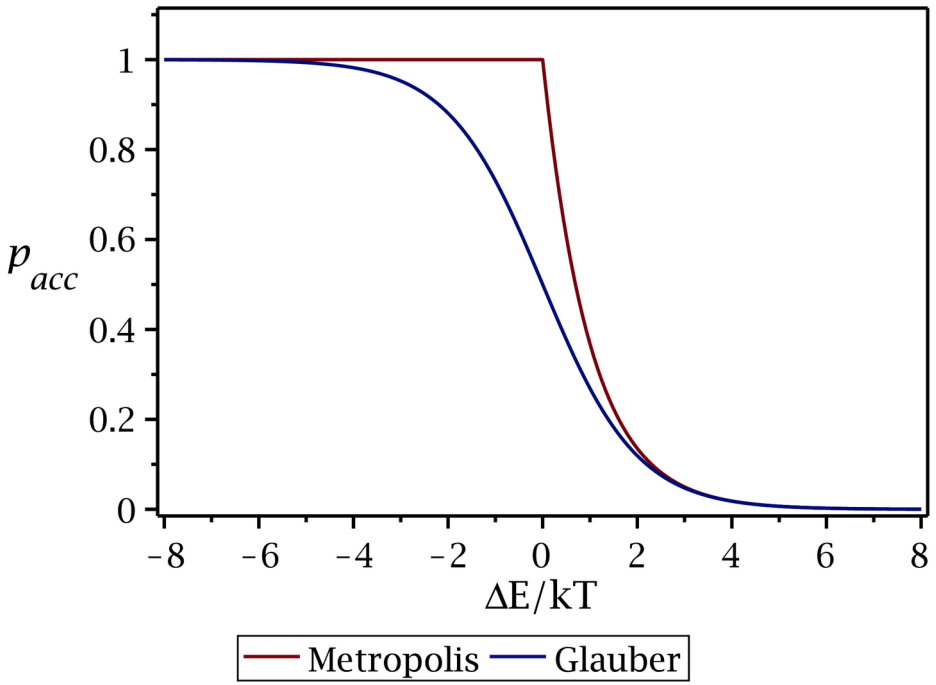
- taka postać p_{acc} również zapewnia spełnienie warunku szczególnej równowagi

Glauber

$$p_{acc}^{G} = \frac{\exp\left(-\frac{E(X)-E(Y)}{kT}\right)}{1+\exp\left(-\frac{E(X)-E(Y)}{kT}\right)} = \frac{1}{1+\exp\left(-\frac{\Delta E}{kT}\right)}$$

Metropolis

$$\mathrm{p}_{acc} = \min\left\{1, \exp\left(-\frac{E(X)-E(Y)}{kT}\right)\right\}$$



- różnica występuje gdy E(X)~E(Y), wpływa to na proces generowania łańcucha Markowa (mniejsze praw. akceptacji) a dokładniej na szybkość dochodzenia układu do równowagi
- dla większych różnic energii oba wzory dają identyczne wyniki
- po osiągnięciu stanu równowagi nie ma znaczenia, którego wzoru użyjemy

Przy użyciu wzoru $T(X_{n+1}|X_n)$ oraz p_{acc} możemy wygenerować ciąg stanów w przestrzeni fazowej tzw. łańcuch Markowa

$$X_1 \to X_2 \to X_3 \to \ldots \to X_n$$

- prawdopodobieństwo wystąpienia stanu X_n zależeć będzie od f(X) i ustalonej temperatury T układu
- w ten sposób nie znając f(X) (bo nie znamy Z) przy użyciu łańcucha Markowa możemy szacować wartości wielkości fizycznych

$$\langle O \rangle = \int_\Omega O(X) f(X) dX \qquad \longrightarrow \qquad \overline{O} = \sum_{i=1}^{n} O(X_i), \qquad X_i \sim dist\{f(X)\}$$

Algorytm Metropolisa generowania łańcucha Markowa

1) aktualna postać łańcucha $\{X_1, X_2, \ldots, X_n\}$

2) wylosuj nowy stan $T(Y|X_n) \to Y$

3) określ prawdopodobieństwo akceptacji nowego stanu $p_{acc} = \min\left\{1, \exp\left(-\frac{E(Y) - E(X_n)}{kT}\right)\right\}$

4) wylosuj liczbę o rozkładzie jednorodnym $U_1 \sim U(0,1)$

5) sprawdź czy wynik jest akceptowalny
$$\begin{cases} U_1 \leq p_{acc} & \to \quad X_{n+1} = Y \\ U_1 > p_{acc} & \to \quad X_{n+1} = X_n \end{cases}$$

- jeśli nie akceptujemy nowego stanu to nowym elementem ciągu jest X_n

6) łańcuch po modyfikacji $\{X_1, X_2, \ldots, X_n, X_{n+1}\}$

- generując odpowiednio długi łańcuch oczekujemy, że dokonamy poprawnego próbkowania f(X) dla danej T inaczej: ciąg stanów będzie ergodyczny (łańcuch aperiodyczny i homogeniczny)
- elementy łańcucha Markowa są skorelowane, bo gdy nie akceptujemy nowego stanu powielamy poprzedni

Metoda symulowanego wyżarzania

Wiemy już jak osiągnąć stan równowagi termodynamicznej w określonej temperaturze - inaczej: otrzymać optymalny rozkład obsadzeń stanów danej temperaturze.

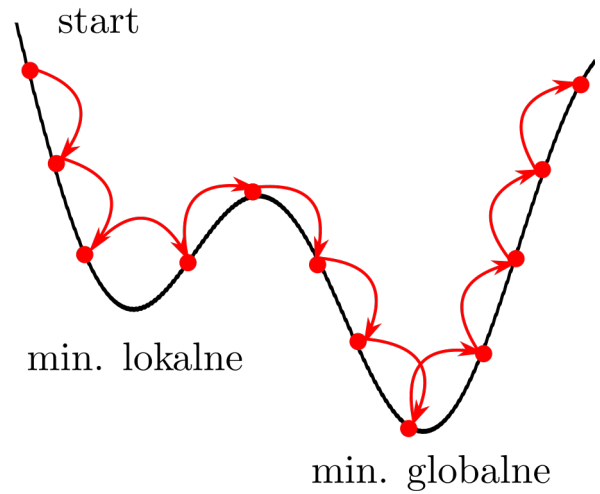
Możemy teraz wykorzystać algorytm Metropolisa w celu znalezienia stanu o najniższej energii, a jeśli energię zamienimy na dowolną inną zmienną lub zestaw zmienych to metoda ta posłuży nam do znalezienia minimum wartości funkcji.

Trzeba tylko umiejętnie dobrać temperaturę....

$$E = f(x)$$

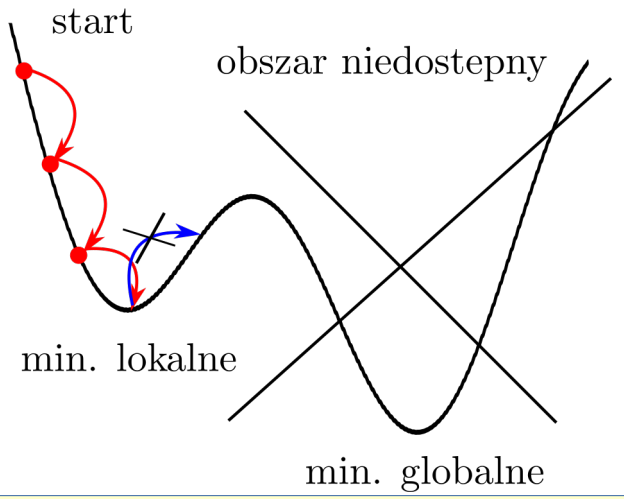
$$\mathrm{p}_{acc} = \min\left\{1, \exp\left(-\frac{f(x_{new}) - f(x_{old})}{kT}\right)\right\}$$

$T >> 1 \rightarrow p_{acc}(E_{new} > E_{old}) > 0$



- jeśli T jest duże, to prawdopodobieństwo akceptacji nowego wyniku też jest duże
- „wędrowiec” może w sposób losowy odwiedzić dowolne miejsce w przestrzeni znajdując minimum

$T << 1 \rightarrow p_{acc}(E_{new} > E_{old}) \sim 0$



- małe T oznacza bardzo małe prawdopodobieństwo zaakceptowania wyniku o większej wartości funkcji celu f(x)
- ruch „wędrowca” ograniczony do lokalnej doliny f(x)
- dostęp do minimum globalnego może zostać zablokowany

Analiza przykładu prowadzi do wniosków

- im wyższa temperatura tym większy obszar dostępny dla wędrowca
- dla wysokiej temperatury wędrowiec przebywa w minimum globalnym tylko przez chwilę i w kolejnej iteracji ucieka
- w niskiej temperaturze ruch wędrowca jest silnie ograniczony przez niskie prawdopodobieństwo akceptacji miejsca w któym funkcja ma większą wartość niż obecnie
- poszukiwanie minimum to proces stochastyczny, aby zwiększyć szansę jego znalezienia można (trzeba?) użyć większej populacji wędrowców
- proces poszukiwania minimum zaczynamy od wysokiej temperatury i stopniowo ją obniżamy, mamy nadzieję że przynajmniej jeden wędrowiec znajdzie się w okolicy minimum globalnego
- konieczne staje się wprowadzenie funkcji opisującej zmiany temperatury, to jeden z kluczowych aspektów algorytmu

Przez analogię do metody powolnego schładzania (wyżarzania) materiałów w celu likwidacji naprężeń i defektów, algorytm ten nosi nazwę metody symulowanego wyżarzania.

Soczewki do największych teleskopów podlegają wyżarzaniu nawet przez 2 lata
– chodzi o zmniejszenie naprężeń które powodują lokalną zmianę współczynnika załamania światła co jest jedną z przyczyn abberacji.

Sposób zmiany temperatury w metodzie SA może być kluczowy dla osiągnięcia minimum.

Najprostszy sposób to zmiana skokowa o zdaną wartość ΔT i wykonywanie IT_{max} iteracji w danej temperaturze w celu osiągnięcia równowagi termodynamicznej z otoczeniem.

inicjalizacja: T_{min}, T_{max}
M - liczba punktow posrednich temperatury
$\Delta T = \frac{T_{max} - T_{min}}{M}$

```
for (m=0; m <= M; m++){

    T = T_max - m * ΔT

    for (it=1; it<=IT_max; it++){
        jeden_krok_SA();
    }

}
```

Można też uzależnić temperaturę od numeru iteracji w sposób bezpośredni bazując na zależności np. jednomianowej

$$y = x^p, \qquad x, y \in (0,1), \quad p \in R$$

$$y = \frac{T - T_{min}}{T_{max} - T_{mni}}$$

$$x = \frac{IT_{max} - it}{IT_{max}}$$

$$T = T(it) = T_{min} + (T_{max} - T_{min}) \left(\frac{IT_{max} - it}{IT_{max}} \right)^p$$

- wyrażenie opisuje temperaturę która w każdej iteracji będzie inna niż w pozostałych
- tempo zmian temperatury zależy od wartości parametru p:
  p=1 - tempo liniowe
  p>1 – tempo nieliniowe, przy czym na początku symulacji temperatura obniża się szybko, później wolniej
  p<1 – tempo nieliniowe, na początku temperatura obniża się wolno, później szybko

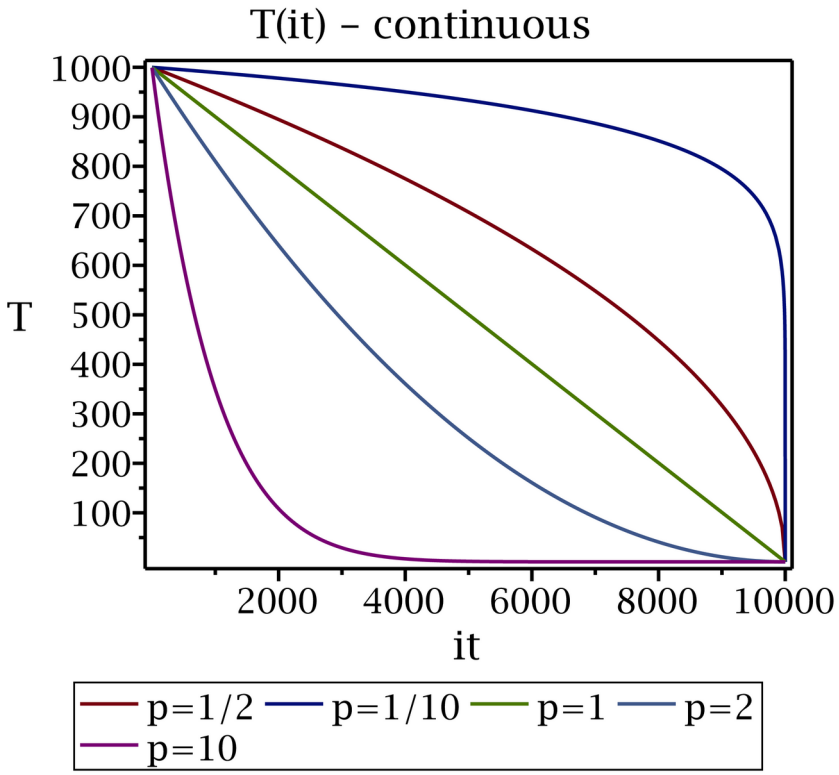
Wzór można dopasować do postaci dyskretnej, co odpowiada wykonywaniu pewnej liczby iteracji w stałej temperaturze

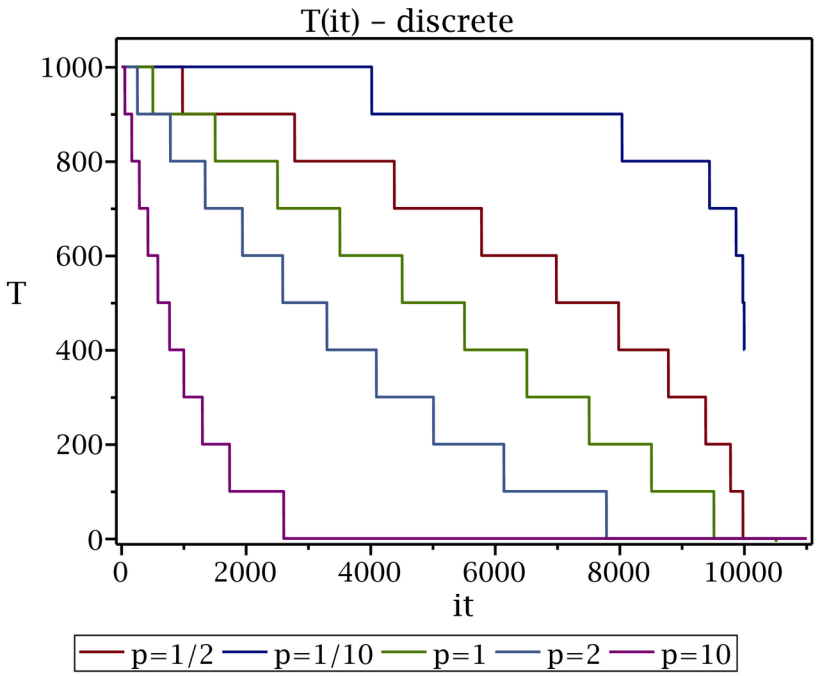
$$\Delta T = \frac{T_{max} - T_{min}}{M}$$

$$T_d = round \left(\frac{T}{\Delta T} \right) \Delta T + T_{min}$$

Przykład – zmiana temperatury w funkcji numeru iteracji

$$T_{min} = 1, \quad T_{max} = 10^3, \quad IT_{max} = 10^4$$





- punktem startowym symulacji może być zmiana liniowa (ciągła lub dyskretna)
- w zależności od przypadku, zmiana liniowa może nie być optymalna,
  czasem warto układ utrzymywać „nieco dłużej” w wyższej temperaturze,
  a później szybko schłodzić – lub odwrotnie,
  przekonać się o tym można wykonując serię symulacji testowych

Algorytm symulowanego wyżarzania

inicjalizacja: T_{min} , T_{max}

M - liczba punktow posrednich temperatury

n – liczba wymiarow

N – liczba wedrowcow

$\vec{\Delta}_{max} = [\Delta_1, \Delta_2, \ldots, \Delta_n]$

$\vec{r}_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,n}]$ – wektor startowy i–tego wedrowca

$\Delta T = \frac{T_{max} - T_{min}}{M}$

for (it =1; it <= IT_{max} ; it ++){

$T = T(it)$

for (i =1; i<= N; i ++){

$\vec{\Delta} \sim dist\{U^n(\cdot,\cdot)\}$ $\qquad |\vec{\Delta}| \leq |\vec{\Delta}_{max}|$

$\vec{r}_i^{new} = \vec{r}_i + \vec{\Delta}$

$p_{acc} = \min\left\{1, \exp\left(-\frac{f(\vec{r}_i^{new}) - f(\vec{r}_i)}{kT}\right)\right\}$

$U_1 \sim U(0,1)$

if ($U_1 \leq p_{acc}$) {

$\vec{r}_i \leftarrow \vec{r}_i^{new}$

} else {

$\vec{r}_i$ – bez zmian

}

}

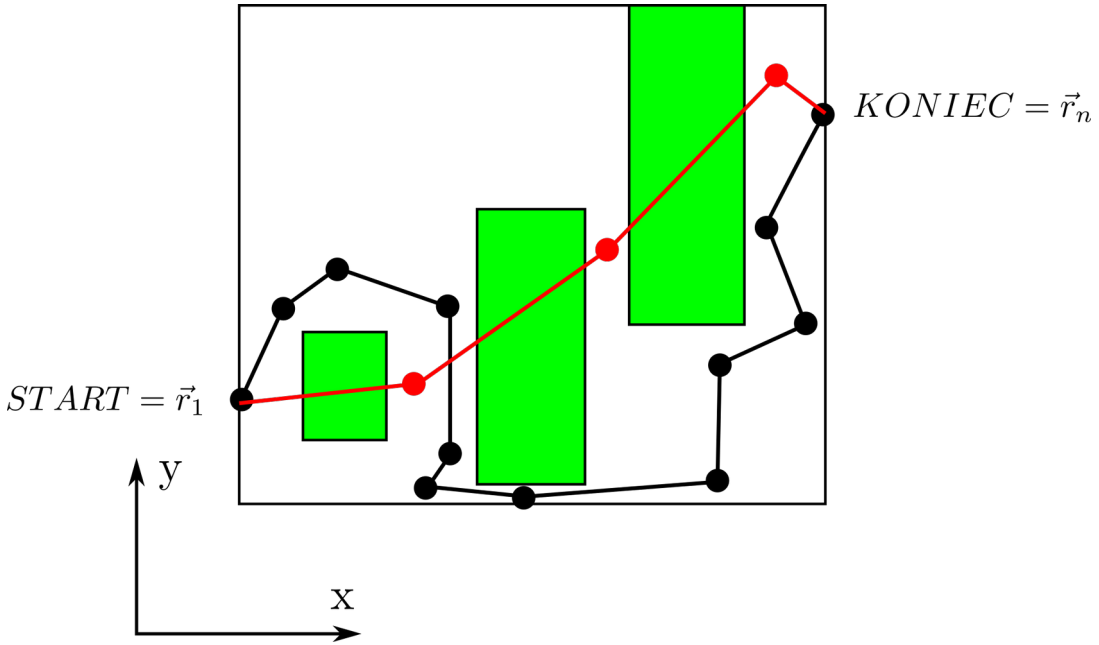
}

Przykład. Poszukiwanie najkrótszej trasy w mieście.

Wykorzystajmy metodę SA do znalezienia najkrótszej drogi łączącej dwa punkty w mieście

Założenia:

- algorytm powinien uwzględniać konieczność omijania budynków
- punkty: startowy i końcowy są ustalone (wejście/wyjście)
- liczba punktów pośrednich określana jest przed symulacją (możemy ją zmieniać, dopasować do problemu)
- położenie punktów pośrednich będziemy zmieniać przy użyciu algorytmu SA - punkty swobodne
- w symulacji użyjemy populacji N niezależnych wędrowców



Uwagi do rysunku:

- obszary zielone to budynki
- czarna krzywa łamana to trasa bliska optymalnej (taką moglibyśmy zaakceptować)
  zawiera dużą liczbę punktów swobodnych
- czerwona krzywa łamana to trasa nieakceptowalna bo przechodzi przez budynki
  zawiera małą liczbę punktów swobodnych
- im więcej krzywa posiada punktów swobodnych tym łatwiej jest jej ominąć budynki, ale jednocześnie zwiększa to liczbę wymiarów opisujących problem

Konstruujemy funkcję celu/kosztu

- argumentami muszą być punkty stanowiące krańce odcinków łączących wejście z wyjściem

$$f = f(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_{n-1}, \vec{r}_n)$$

$\vec{r}_1, \vec{r}_n$ - parametry modelu

$\vec{r}_2, \vec{r}_3, \ldots, \vec{r}_{n-1}$ - zmienne (stopnie swobody)

ograniczenia przestrzenne $x_i \in [x_{min}, x_{max}]$
$y_i \in [y_{min}, y_{max}]$

wymiarowość problemu $dim\{\vec{r}_2, \vec{r}_3, \ldots, \vec{r}_{n-1}\} = 2 \cdot (n-1)$

Ponieważ trasa składa się z odcinków, więc najłatwiej skonstruować funkcję celu w postaci sumy wkładów wnoszonych przez poszczególne odcinki

$$f = \sum_{i=1}^{n-1} d_{i,i+1}$$

- jeśli trasa przechodzi pomiedzy budynkami to f określa jej długość
- najmniejsza wartość f powinna opisywać najkrótszą trasę
- jeśli odcinek przecina budynek to oprócz jego długości współczynnik $d_{i,i+1}$ powienien zawierać dodatkowy nieujemny czynnik stanowiący karę,
  to spowoduje że wartość funkcji celu bardzo wzrośnie i takie rozwiązanie będzie odrzucane lub akceptowane z małym prawdopodobieństwem

- wkład danego odcinka z możliwą karą za przejście przez budynek np. w postaci potencjału schodkowego

$$d_{i,i+1} = \int_{\vec{r}_i}^{\vec{r}_{i+1}} [1 + V(\vec{r})]\, ds, \qquad ds = \sqrt{dx^2 + dy^2}$$

$$V(\vec{r}) = \begin{cases} 0, & \text{empty field} \\ V_{max} >> 1, & \text{building} \end{cases}$$

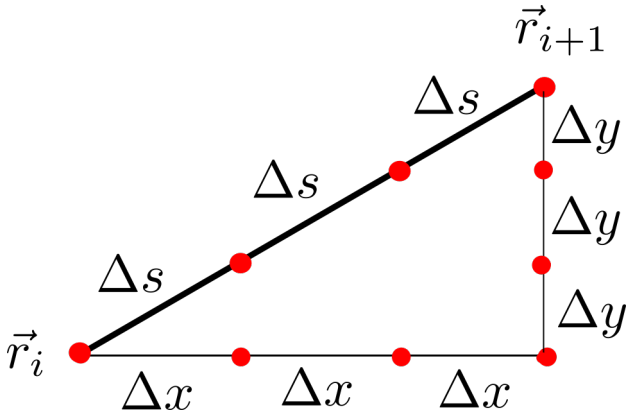
Całkowanie wykonajmy numerycznie przy użyciu metody trapezów dla (K+1) węzłów
- takie podejście jest dość ogólne i pozwala na łatwą zmianę potencjału kary

$$\Delta x = \frac{x_{i+1} - x_i}{K} \qquad \Delta y = \frac{y_{i+1} - y_i}{K}$$

$$x_\mu = x_i + \mu \cdot \Delta x, \qquad \mu = 0, 1, \ldots, K$$

$$y_\mu = y_i + k \cdot \Delta y, \qquad \mu = 0, 1, \ldots, K$$

$$\vec{r}_\mu = [x_\mu, y_\mu]$$



$$d_{i,i+1} = \int_{\vec{r}_i}^{\vec{r}_{i+1}} [1 + V(\vec{r})]\, ds \approx \sum_{\mu=0}^{K} c_\mu \left[V(\vec{r}_\mu)\right] \cdot \Delta s$$

$$\Delta s = \sqrt{\Delta x^2 + \Delta y^2}$$

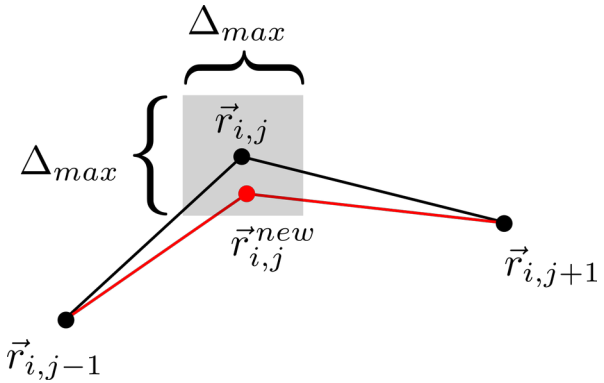
$$c_\mu = \begin{cases} \frac{1}{2} & \Longleftrightarrow \quad \mu = 0, K \\ 1 & \Longleftrightarrow \quad \mu = 2, 3, \ldots, K-1 \end{cases}$$

Uwaga: spodziewamy się że tylko część odcinka Δs może przechodzić przez budynek, co prowadzi do powstawiania błędu całkowania, ale jeśli K jest duże (np. K=100) to błąd ten staje się mały, pamietajmy także że naszym celem jest trasa omijająca budynki – wówczas błąd całkowania będzie znikał (bo potencjał kary znika)

- konstruujemy tablicę danych zawierającą informacje o trasach N wędrowców k - w niej będziemy dokonywać zmian

$$\vec{f} = \begin{array}{|c|c|c|c|c|c|} \hline \vec{r}_1 & \vec{r}_{1,2} & \vec{r}_{1,3} & \dots & \vec{r}_{1,n-1} & \vec{r}_n \\ \hline \vec{r}_1 & \vec{r}_{2,2} & \vec{r}_{2,3} & \dots & \vec{r}_{2,n-1} & \vec{r}_n \\ \hline \vec{r}_1 & \vdots & \vdots & \vdots & \vdots & \vec{r}_n \\ \hline \vec{r}_1 & \vec{r}_{N,2} & \vec{r}_{N,3} & \dots & \vec{r}_{N,n-1} & \vec{r}_n \\ \hline \end{array}$$

- zmiany położeń punktów swobodnych wykonujemy losowo w obu kierunkach nie przekraczając wyznaczonych granic obszaru Ω



Δ_{max} – ustalone
$U_1, U_2 \sim U(0,1)$
$\delta x = (2U_1 - 1)\Delta_{max}$
$\delta y = (2U_2 - 1)\Delta_{max}$
$\vec{r}^{new}_{i,j} = \vec{r}_{i,j} + [\delta x, \delta y]$
$\Omega = [x_{min}, x_{max}] \times [y_{min}, y_{max}]$
if $(\vec{r}^{new}_{i,j} \in \Omega)\{$
$p_{acc} = \min\left\{1, \exp\left(-\beta[f(\vec{r}^{new}_{i,j}) - f(\vec{r}_{i,j})]\right)\right\} = \min\left\{1, \exp\left(-\beta[d^{new}_{j-1,j} + d^{new}_{j,j+1} - d_{j-1,j} - d_{j,j+1}]\right)\right\}$
$U_3 \sim U(0,1)$
if $(U_3 \leq p_{acc})\{$
$\vec{r}_{i,j} \leftarrow \vec{r}^{new}_{i,j}$
$\}$ else $\{$
$\vec{r}_{i,j}$ – bez zmian
$\}$
$\}$

$$d_{j,j+1} = \int_{\vec{r}_j}^{\vec{r}_{j+1}} [1 + V(\vec{r})]\, ds$$

Optymalizacja Monte Carlo

Parametry symulacji:

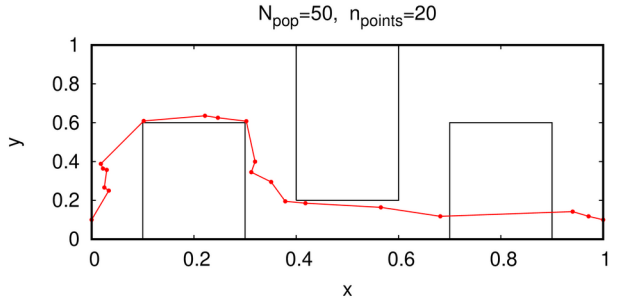
$$\beta_{min} = 10^{-6}, \quad \beta_{max} = 10^2$$

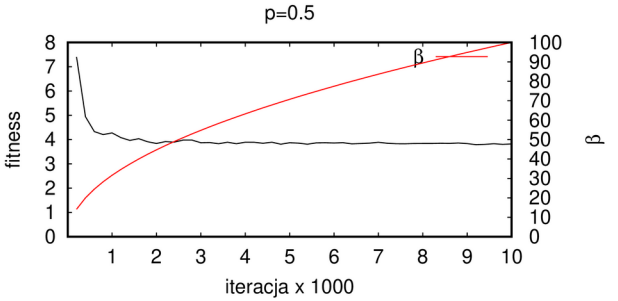
$$N = 50, \quad n = 20, \quad IT_{max} = 10^4$$

$$\Omega = [0,1] \times [0,1]$$

$$\Delta_{max} = 0.03$$

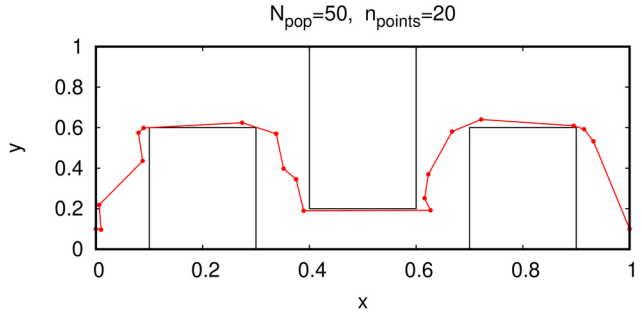
$$p = \frac{1}{2}, 1, 2, 6$$

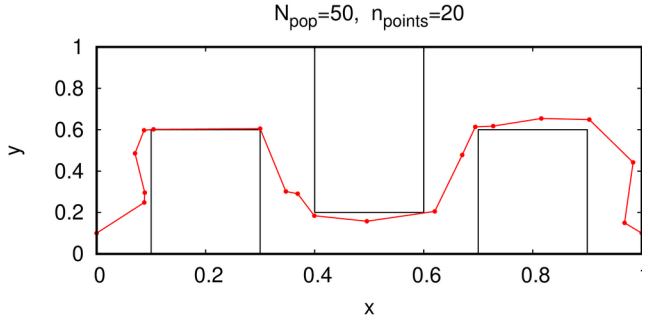


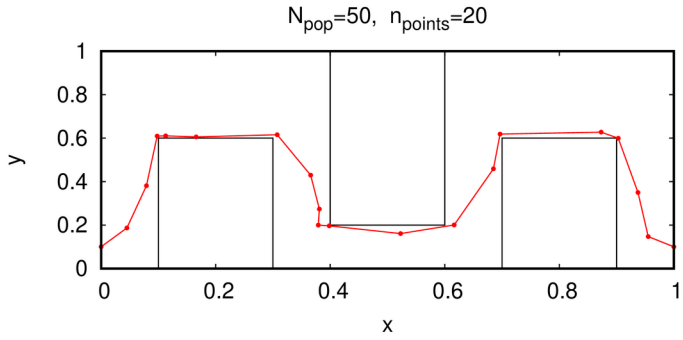


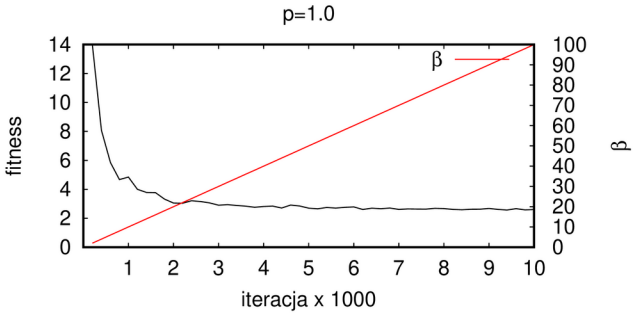
tu optymalizacja ścieżki nie powiodła się, ponieważ wartość β rośnie zbyt szybko co obniża p_{acc}, a trzeba przesunąć odcinek (2 punkty)

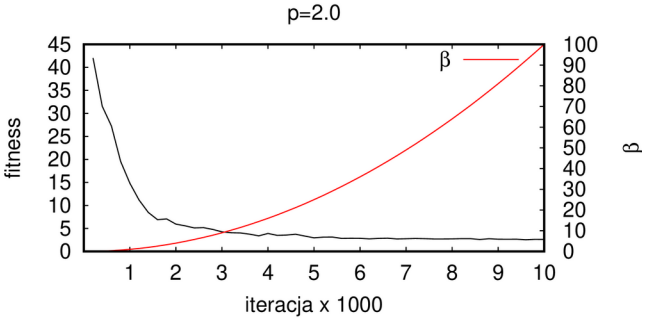
$$f_{p=1/2} = 3.82$$

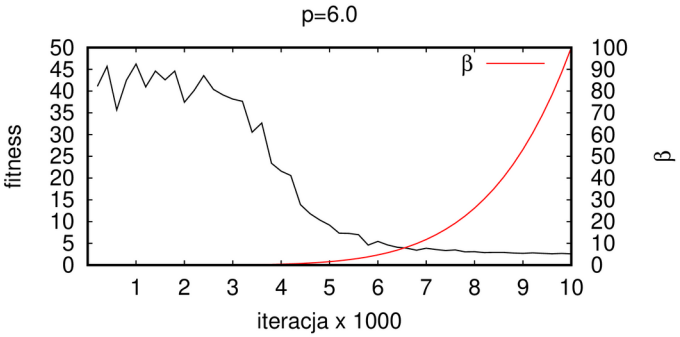












$$f_{p=1} = 2.601$$

$$f_{p=2} = 2.617$$

$$f_{p=6} = 2.575$$