Základy umělé inteligence

Algoritmy iterativní optimalizace, Populační metody

Ing. Tomáš Řehořek

Computational Intelligence Group (CIG),
Katedra teoretické informatiky (KTI),
Fakulta informačních technologií (FIT),
České vysoké učení technické v Praze (ČVUT)

BI-ZUM, LS 2016/17, 3. přednáška

https://edux.fit.cvut.cz/courses/BI-ZUM/

Minulá přednáška

- Algoritmy hledání **cesty** z počátečního do cílového stavu ve stavovém prostoru
  - Dijkstrův algoritmus,
  - Hladové prohledávání,
  - A*,
- Pro mnohé problémy je cesta irelevantní, důležitý je pouze cílový **stav**
  - Problém N dam,
  - Symbolická integrace bez důkazu,
  - Optimalizace vektoru z $\mathbb{R}^n$,
  - Optimalizace vektoru z $\{0, 1\}^n$
- Dnešní přednáška: algoritmy **iterativní optimalizace**
  - Práce pouze s kandidujícím koncovým stavem a jeho postupné zlepšování

Optimalizační problém

Optimalizační problém: Zjednodušená definice

Nechť X je libovolná množina, kterou budeme nazývat **množina přípustných řešení**, a f je zobrazení $f\colon X \to \mathbb{R}$, které nazýváme **kriteriální funkce**.

Optimalizační problém je pak formulován jako hledání $\mathbf{x}^* \in X$ maximalizujícího funkční hodnotu $f(\mathbf{x})$:

$$\mathbf{x}^* \in \arg\max_{\mathbf{x}\in X} f(\mathbf{x})$$

Zapisujeme jako:

$$\underset{\mathbf{x}\in X}{\text{maximize}} \quad f(\mathbf{x})$$

V praxi není podstatné, zda funkční hodnotu maximalizujeme či minimalizujeme. Platí totiž

$$\arg\max_{\mathbf{x}\in X}(-f(\mathbf{x})) = \arg\min_{\mathbf{x}\in X}(f(\mathbf{x})).$$

Příklady optimalizačních problémů

Optimalizace sázkového / investičního portfolia

Sázková kancelář vypsala kurzy na různé výsledky zápasu:

Výsledek	1	10	0	02	2
Kurz	1.27	1.02	4.7	3.09	9.00

1 … výhra domácích
10 … výhra domácích nebo remíza
0 … remíza
02 … výhra hostů nebo remíza
2 … výhra hostů

Kancelář umožňuje uzavírat sázky na libovolnou kombinaci událostí. V rámci marketingové kampaně dává kancelář 50% bonus, a to až do výše 1 000 Kč, tedy např. při sázce 2 000 Kč smíme vsadit celkem 3 000 Kč.

Je možné vsadit tak, abychom měli zaručen minimální zisk, a to bez ohledu na výsledek zápasu? Jak sázku rozdělit mezi jednotlivé možné výsledky?

Příklady optimalizačních problémů

Optimalizace sázkového / investičního portfolia

Formalizace problému:

Výsledek	1	10	0	02	2
Kurz	$x_1 \cdot 1.27$	$x_2 \cdot 1.02$	$x_3 \cdot 4.7$	$x_4 \cdot 3.09$	$x_5 \cdot 9.0$

Množina přípustných řešení:

$$X = \left\{ (x_1, \ldots, x_5) \in \mathbb{R}^5 \,\middle|\, \sum_{i=5}^{5} x_i = 3000 \right\}$$

Kriteriální funkce:

$$f : (x_1, \ldots, x_5) \mapsto \min\left(\{1.27x_1+1.02x_2, 1.02x_2+4.7x_3+3.09x_4, 3.09x_4+9.0x_5\}\right)$$

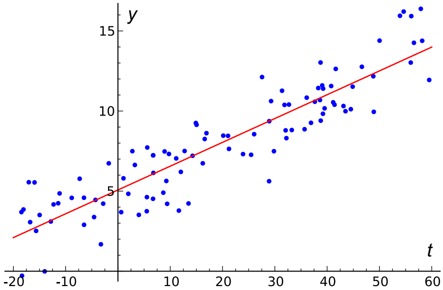
Příklady optimalizačních problémů

Lineární regrese

Je dána množina měření

$$Y = \{(t_1, y_1), (t_2, y_2), \ldots, (t_n, y_n)\} \subseteq \mathbb{R}^2$$

Naším úkolem je nalézt funkci ℓ ve tvaru $\ell(t) = a \cdot t + b$ minimalizující sumu čtverců odchylek mezi $\ell(t_i)$ a y_i.



Množina přípustných řešení:

$$X = \mathbb{R}^2$$

Kriteriální funkce:

$$f : (a, b) \mapsto \sum_{i=1}^{n} (a \cdot t_i + b - y_i)^2$$

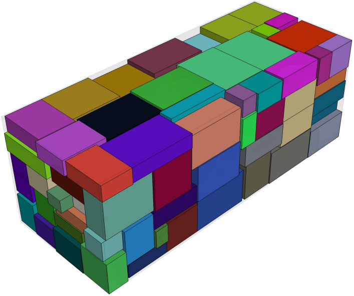
Příklady optimalizačních problémů

Plnění kontejneru

Je dána množina krabic tvaru kvádru, přičemž každá krabice je popsána různými charakteristikami:

- rozměry,
- hmotnost,
- nosnost...

Naším cílem je co nejvíce naplit kontejner krabicemi, a to s ohledem na stabilitu, nosnost krabic umístěných pod jinými krabicemi atd.



Množina přípustných řešení: Obsahuje platné konfigurace umístění krabic do kontejneru (prázdný průnik mezi každými dvěma krabicemi, stabilita, ohled na nosnost).

Kriteriální funkce: Suma objemů krabic použitých v dané konfiguraci.

Příklady optimalizačních problémů

Další průmyslové problémy

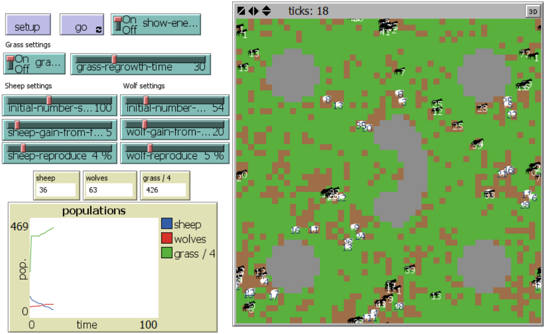
Další průmyslové optimalizační problémy:

- přidělení služeb zaměstnancům,
- rozvoz zboží,
- návrh elektrických obvodů,
- osazování plošných spojů,
- dělení ocelových odlitků,
- návrh křižovatek ve městě,
- tvorba univerzitního rozvrhu. . .

Příklady optimalizačních problémů

Řídící algoritmus robota

Je dáno prostředí, v němž se vyskytují ovce, vlci a tráva. Máme pod kontrolou ovce, nikoli však vlky ani trávu. Při kontaktu s vlkem ovce zahyne. Ovce i vlci mají energii, která klesá s časem a doplňuje se potravou. Potravou vlků jsou ovce, potravou vlků je tráva. Tráva postupně doůstá. Ovce mají senzory, jimiž dokáží detekovat blízké vlky a jejich vzdálenost. Při dostatku energie se ovce i vlci rozmnožují (duplikují). Celé prostředí je řízeno řadou netriviálních konstant.



Naším úkolem je nalézt řídící program ovce, který v každém kroku simulace vrátí úhel $\alpha \in \langle 0, 2\pi)$ a vzdálenost $d \in \langle 0, 1\rangle$, kterou má ovce ujít. Tento program má, bude-li sdílen všemi ovcemi, maximalizovat počet ovcí vyskytujících se v prostředí po 1000 krocích simulace.

Formalizace optimalizačních problémů

Matematická optimalizace

Optimalize je rozsáhlá matematická disciplína s celou řadou formálních aparátů pro formulaci optimalizačních problémů.

Naše definice množiny X přípustných řešení je často nepraktická a rozděluje se na dva kroky:

1. specifikace prostoru $\hat{X}$, do nějž řešení náleží,
2. specifikace **omezujících podmínek** indukujících množinu přípustných řešení $X \subseteq \hat{X}$

Dominantní roli má prostor $\hat{X} = \mathbb{R}^n$, resp. $\hat{X} = \mathbb{Z}^n$

- do konce dnešní přednášky budeme uvažovat pouze tyto prostory,
- prostory stromů, stavových automatů: příští přednáška

Matematický optimalizační problém

Definice

Nechť f, $g_1, \dots, g_J$ a $h_1, \dots, h_K$ jsou libovolné funkce $\mathbb{R}^n \to \mathbb{R}$.

Matematický optimalizační problém maximalizující f s omezujícími podmínkami $g_1, \dots, g_J$ a $h_1, \dots, h_K$ je hledání $\mathbf{x}^* \in \mathbb{R}^n$ minimalizujícího rozdíl

$$\left(\max_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})\right) - f(\mathbf{x}^*)$$

při splnění nerovností $g_j(\mathbf{x}) \geq 0$ a $h_k(\mathbf{x}) = 0$.

Standardní zápis:

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{maximize}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g_j(\mathbf{x}) \geq 0,\ j \in \{1, \dots, J\}, \\ & h_k(\mathbf{x}) = 0,\ k \in \{1, \dots, K\}. \end{aligned}$$

Třídy optimalizačních problémů

Existují **speciální třídy** „snadných“ optimalizačních problémů, pro které jsou známy spolehlivé a rychlé algoritmy:

- **nejmenší čtverce** (Least-squares)
  - minimalizace $\|\mathbf{Ax} - \mathbf{b}\|_2^2 = \sum_{i=1}^{m} (\mathbf{a}_i \cdot \mathbf{x} - b_i)^2$,
  - řeší např. problém lineární regrese,
- **lineární programování** (Linear programming)
  - minimalizace $\mathbf{c}^T\mathbf{x}$ při $\mathbf{a}_i^T\mathbf{x} \leq b_i$ pro $i \in \{1, \ldots, m\}$,
  - řeší např. problém optimalizace sázkového portfolia,
- **konvexní optimalizační problém**.

Obecný optimalizační problém je však **velmi obtížně řešitelný**

- kriteriální funkce i omezující podmínky mohou být velmi složité,
- právě zde nastupuje **umělá inteligence!**

Obecný optimalizační problém: Algoritmy AI

Specializované optimalizační algoritmy (nejmenší čtverce, lineární programování) pracují se **známou vnitřní strukturou** problému

- omezující podmínky i kriteriální funkce mají známý a standardní tvar

Obecný optimalizační problém může mít libovolně komplexní strukturu

- NP-úplné optimalizační problémy

Struktura obecného optimalizačního problému navíc nemusí být dobře formalizovatelná či dokonce známá

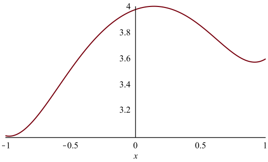
- úspěšnost řídící jednotky robota v simulovaném prostředí

Algoritmy iterativní optimalizace

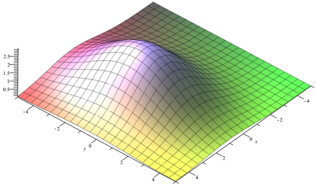
- metody AI, které se snaží řešit obecný optimalizační problém metodou „pokus-omyl“,
- typicky získávají znalosti o struktuře problému v průběhu optimalizace a s těmito znalostmi pracují,
- téma zbytku dnešní přednášky

Optimalizace vektoru z $\mathbb{R}^n$

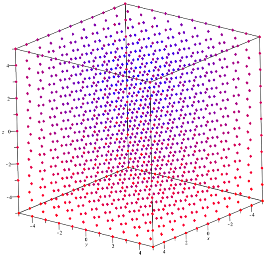
Možnosti vizualizace funkce pro $n = 1, 2, 3$



1D funkce,
2D graf



2D funkce,
3D graf



3D funkce,
4D graf

Algoritmy iterativní optimalizace: Obecné schéma

Jelikož předpokládáme, že o funkci f nic nevíme, nelze použít optimalizační metody z matematické analýzy (hledání bodů s nulovou derivací atd.)

Připadá však v úvahu následující iterativní schéma:

Algorithm 1 Obecná iterativní optimalizace

x $\leftarrow$ náhodně vygenerované počáteční řešení
while x není dost dobré $\wedge$ algoritmus neběží moc dlouho **do**
    y $\leftarrow$ nový kandidát
    if $f(\mathbf{y}) > f(\mathbf{x})$ **then**
        $\mathbf{x} \leftarrow \mathbf{y}$
    end if
end while
return x

Optimalizace hrubou silou

- prostor navzorkujeme systematicky a vrátíme nejlepší řešení
- v praxi nelze použít, zejm. pro prostory velké dimenze

Příklad pro optimalizaci v hyperkrychli $\langle 0, 10\rangle \times \langle 0, 10\rangle \times \ldots \times \langle 0, 10\rangle$:

Algorithm 2 Brute-force optimization on $\langle 0, 10\rangle \times \langle 0, 10\rangle \times \ldots \times \langle 0, 10\rangle$

```
x ← (0, 0, ..., 0)
for all y_1 ∈ {0, 0.1, 0.2, ..., 9.9, 10} do
    for all y_2 ∈ {0, 0.1, 0.2, ..., 9.9, 10} do
        ⋮
        for all y_n ∈ {0, 0.1, 0.2, ..., 9.9, 10} do
            if f((y_1, y_2, ..., y_n)) > f(x) then
                x ← (y_1, y_2, ..., y_n)
            end if
        end for
        ⋮
    end for
end for
```

Náhodná optimalizace

- optimalizace hrubou silou je často výpočetně nezvládnutelná,
- náhodná optimalizace vzorkuje menší množství náhodně generovaných řešení

Příklad pro optimalizaci v hyperkrychli $\langle 0, 10\rangle \times \langle 0, 10\rangle \times \ldots \times \langle 0, 10\rangle$:

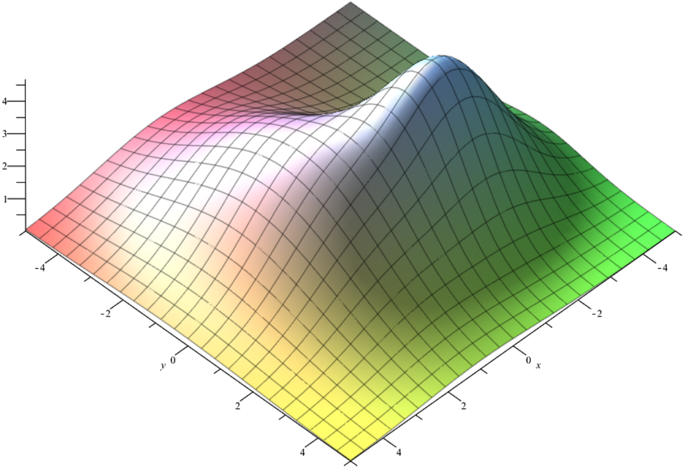
Algorithm 3 Random optimization on $\langle 0, 10\rangle \times \langle 0, 10\rangle \times \ldots \times \langle 0, 10\rangle$

$\mathbf{x} \leftarrow (0, 0, \ldots, 0)$
for $i \leftarrow 1 \ldots max\_steps$ **do**
  $\mathbf{y} \leftarrow (\mathrm{random}(0, 10), \mathrm{random}(0, 10), \ldots, \mathrm{random}(0, 10))$
  if $f(\mathbf{y}) > f(\mathbf{x})$ **then**
    $\mathbf{x} \leftarrow \mathbf{y}$
  end if
end for
return $\mathbf{x}$

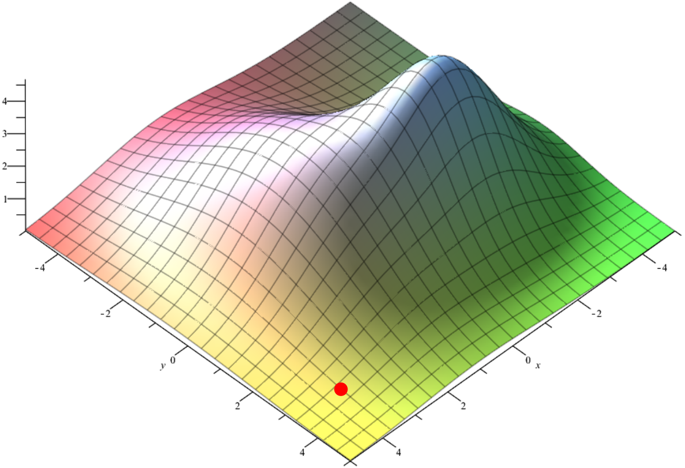
Hill climbing

- Random a brute-force optimalizace jsou neefektivní
  - trvají dlouho,
  - nereflektují informace o funkci, které v průběhu optimalizace o funkci získávají
- **Hill climbing** využívá dosud nejlepšího řešení $\mathbf{x}$:
  - generuje body v jeho **okolí** a pokud je vygenerovaný bod lepší, provede náhradu,
  - analogie se šplháním do kopce: rozhlížíme se a jdeme nahoru
- Co znamená generování v okolí nějakého řešení?
  - pro vektor $\mathbf{x} \in \mathbb{R}^n$ např. jiný vektor $\mathbf{y}$ vzálený maximálně ε od $\mathbf{x}$, tj. $\|\mathbf{x} - \mathbf{y}\| \leq \varepsilon$,
  - pro binární vektor z $\{0, 1\}^n$ jiný vektor lišící se max. v k bitech,
  - pro problém N dam konfigurace šachovnice lišící se v pozici jedné dámy,
  - obecně volba nějakého sousedního uzlu ve stavovém prostoru

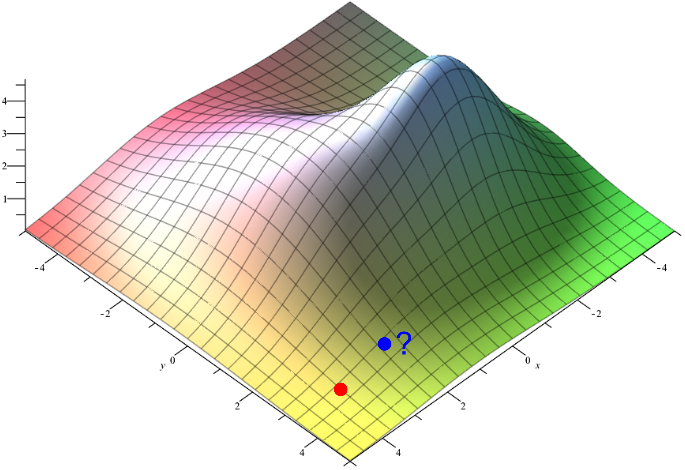
Hill climbing: Ilustrace



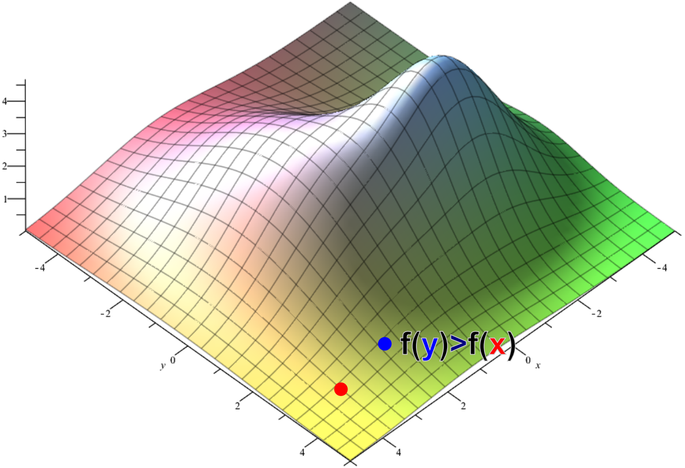
Hill climbing: Ilustrace



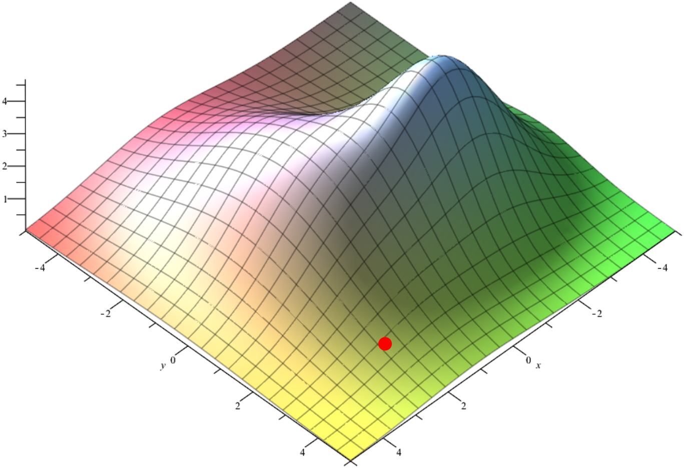
Hill climbing: Ilustrace



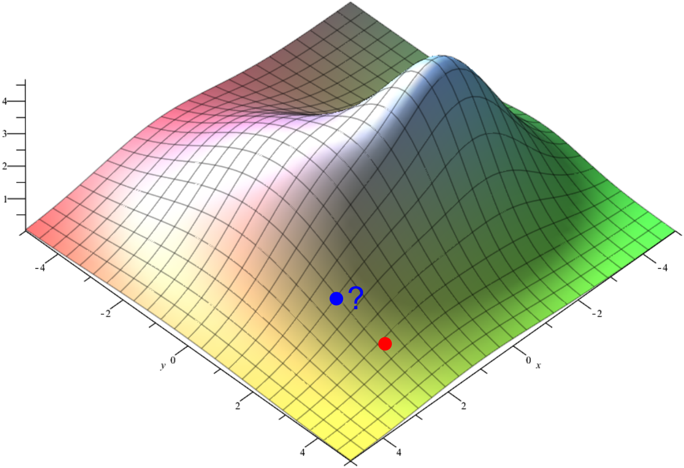
Hill climbing: Ilustrace



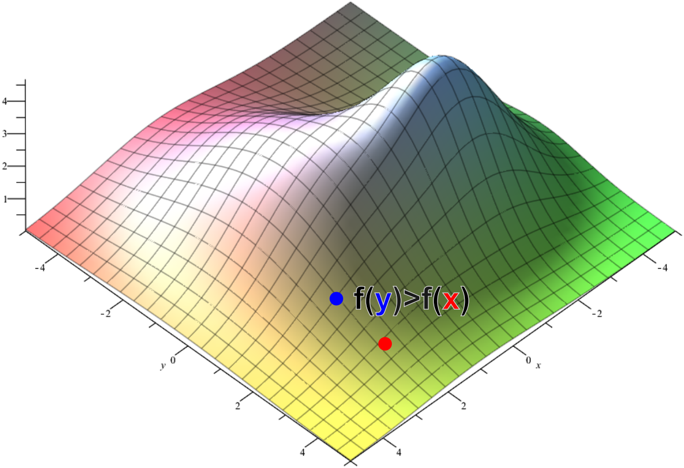
Hill climbing: Ilustrace



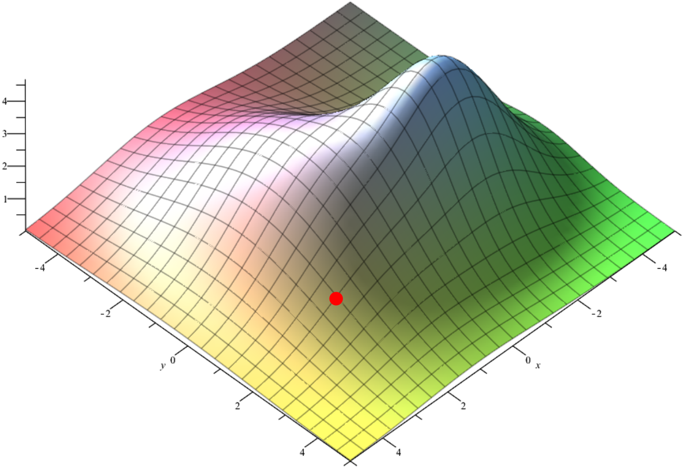
Hill climbing: Ilustrace



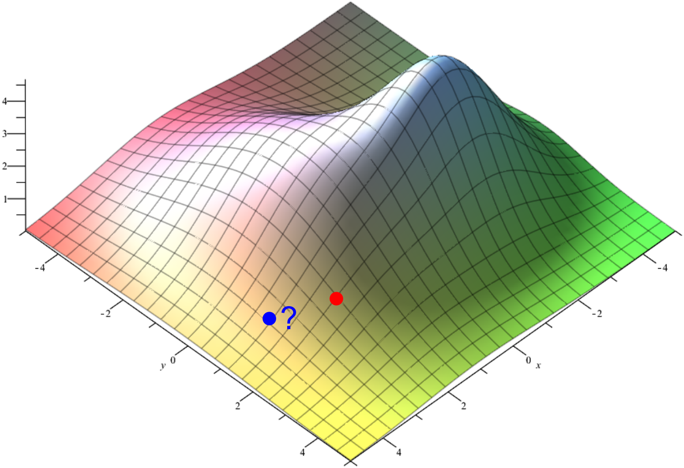
Hill climbing: Ilustrace



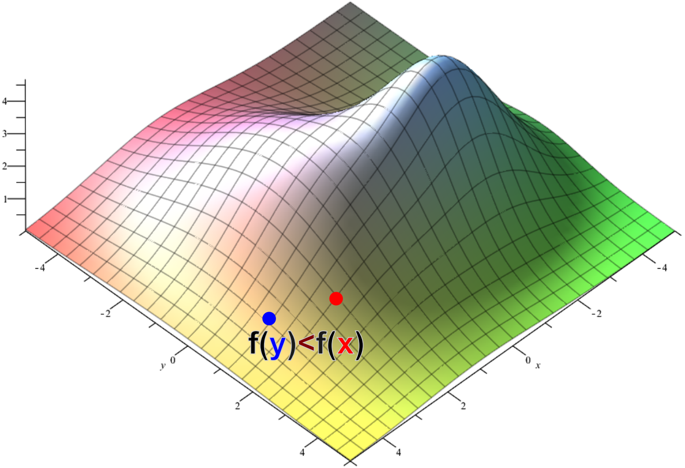
Hill climbing: Ilustrace



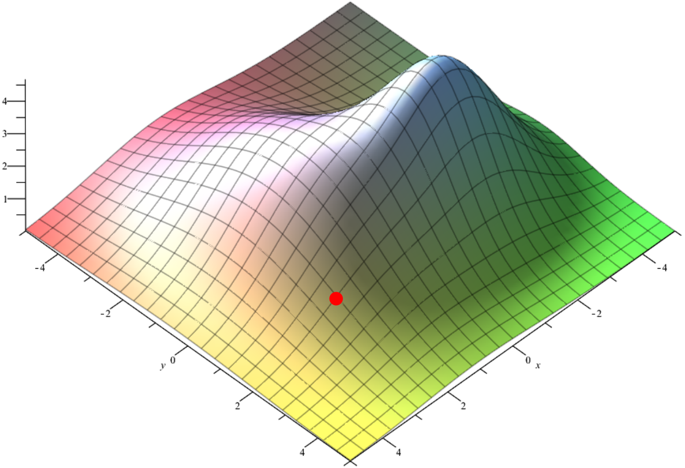
Hill climbing: Ilustrace



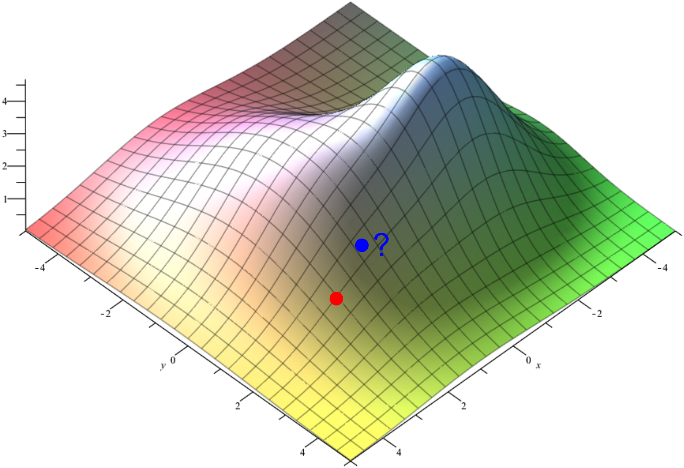
Hill climbing: Ilustrace



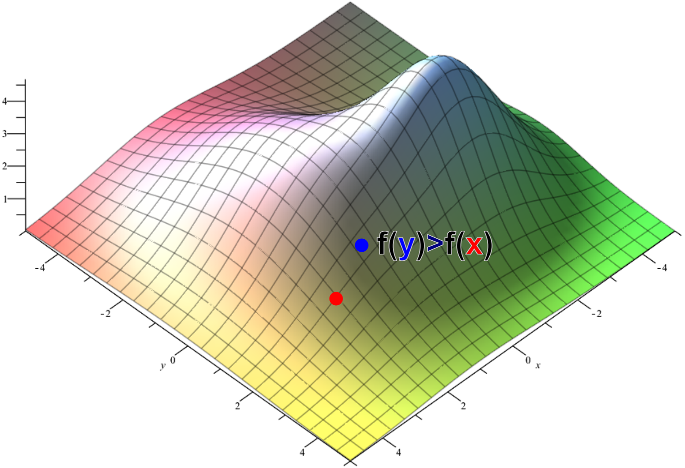
Hill climbing: Ilustrace



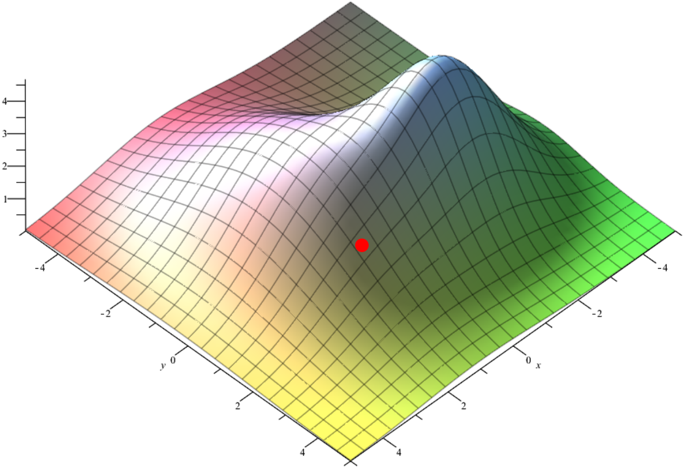
Hill climbing: Ilustrace



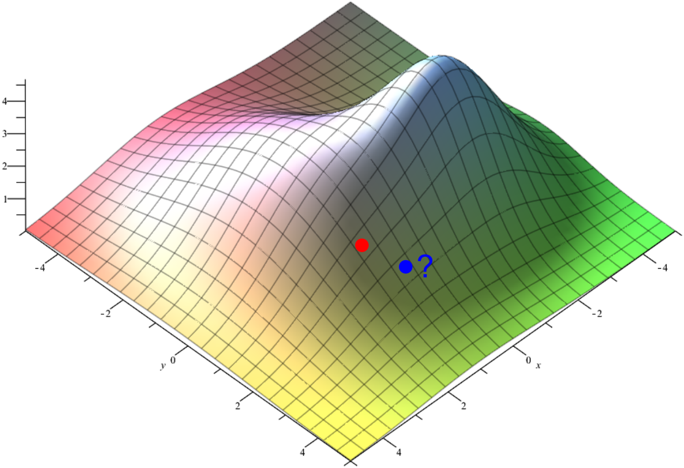
Hill climbing: Ilustrace



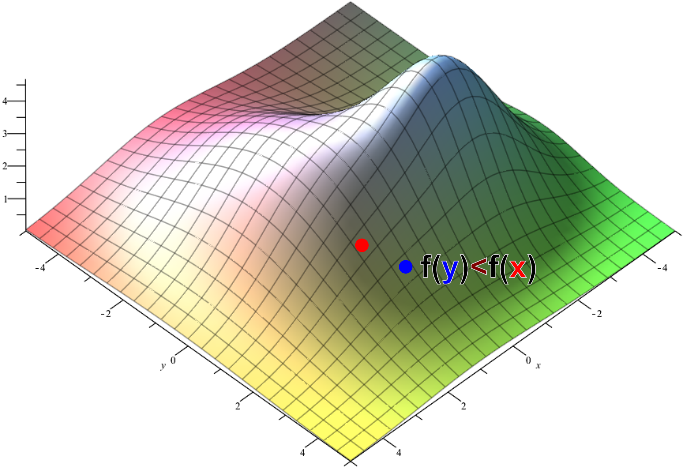
Hill climbing: Ilustrace



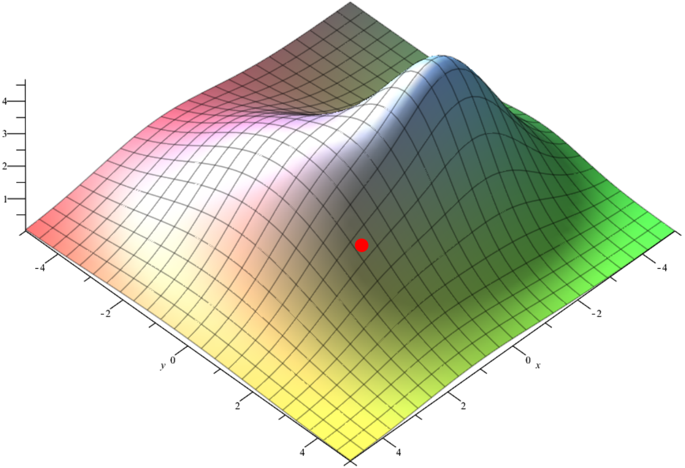
Hill climbing: Ilustrace



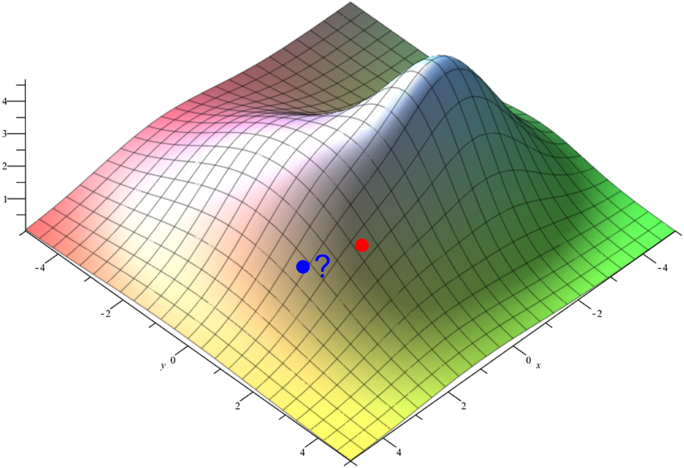
Hill climbing: Ilustrace



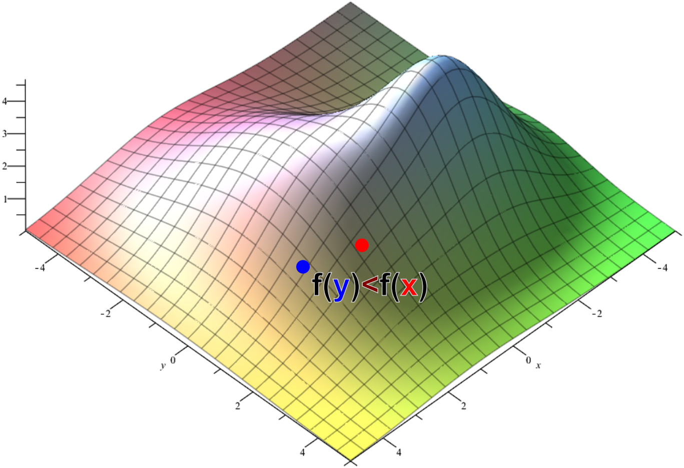
Hill climbing: Ilustrace



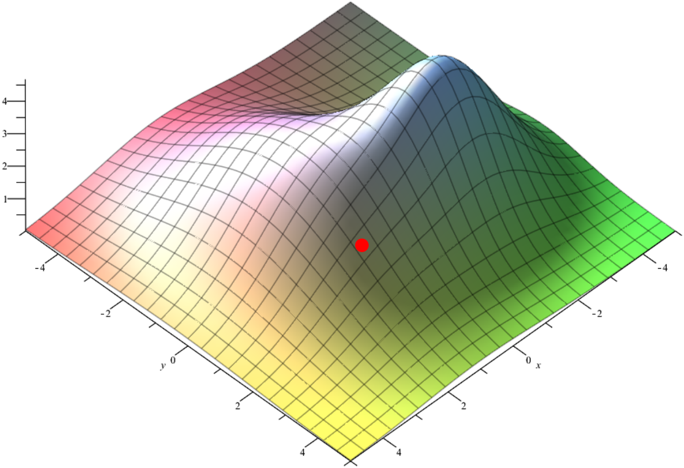
Hill climbing: Ilustrace



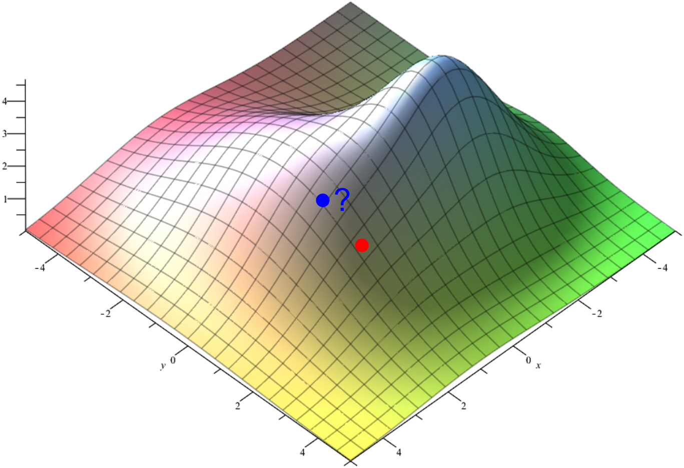
Hill climbing: Ilustrace



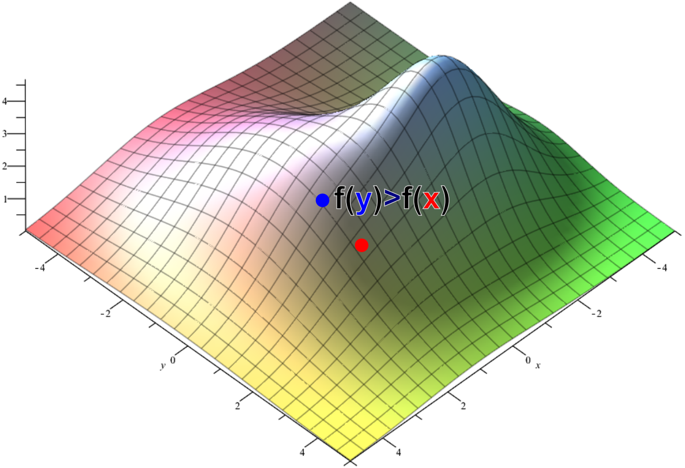
Hill climbing: Ilustrace



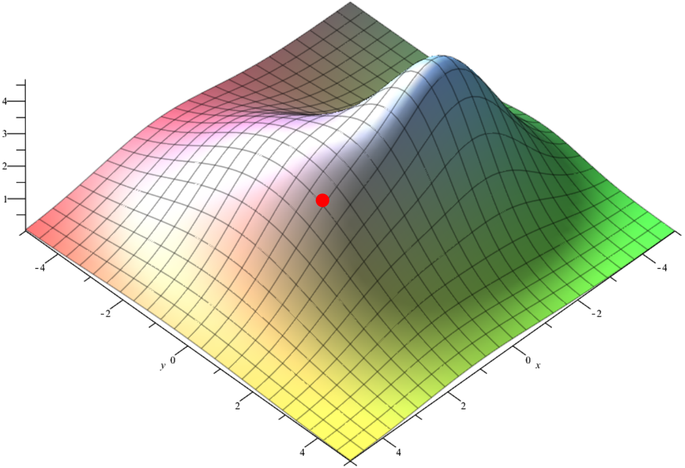
Hill climbing: Ilustrace



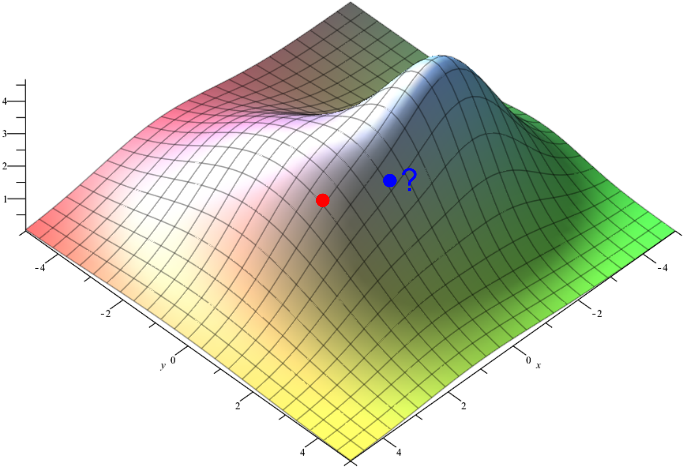
Hill climbing: Ilustrace



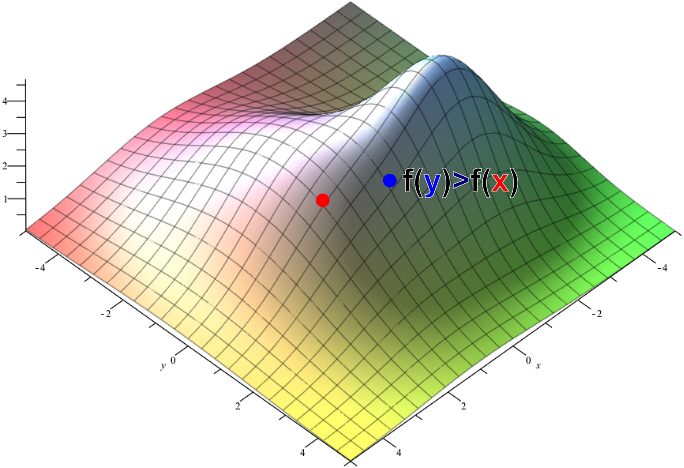
Hill climbing: Ilustrace



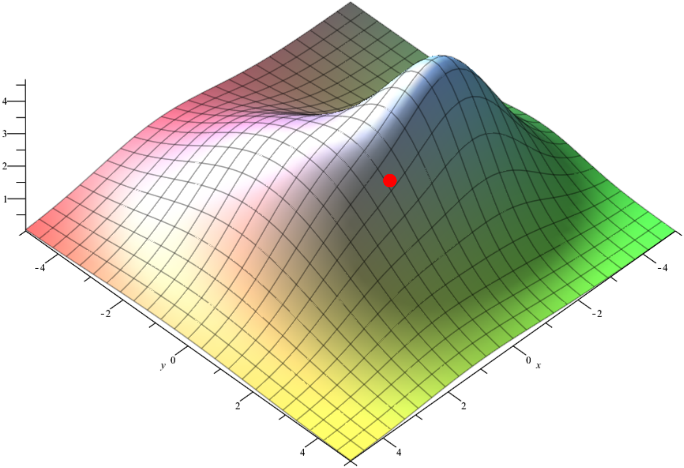
Hill climbing: Ilustrace



Hill climbing: Ilustrace



Hill climbing: Ilustrace



Hill climbing: Pseudokód

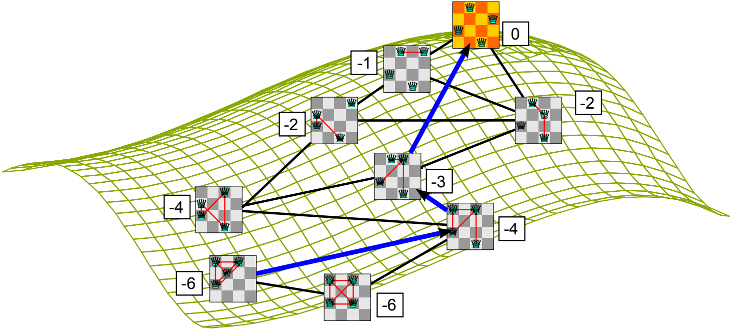
Algorithm 4 Hill climbing

```
x ← random_state()
i ← 0
while ¬good_enough(x) ∧ i < max_iter do
    y ← random_neighbor(x)
    if f(y) > f(x) then
        x ← y
    end if
    i ← i + 1
end while
return x
```

Hill climbing: Ilustrace pro problém *N* dam

Iterativní optimalizace se zdaleka netýká jen vektorů z $\mathbb{R}^n$!

Obrázek (pozor, zavádějícím způsobem) ukazuje, že Hill climbing lze použít i na jiných než numerických stavových prostorech.



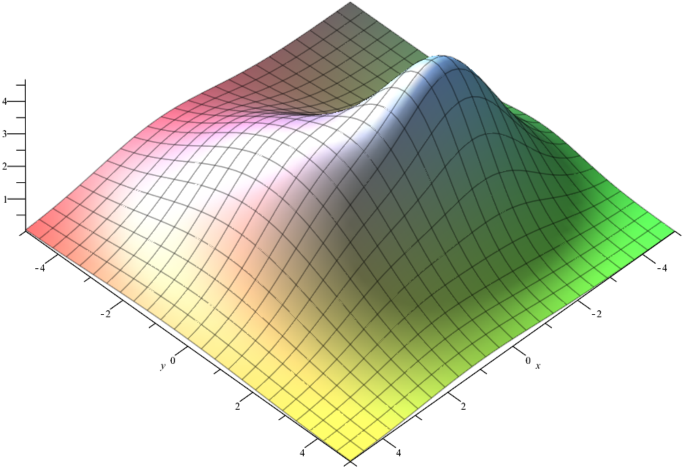
Steepest ascent Hill climbing

Varianta Hill climbingu, která v každém kroku generuje více kandidujících sousedů a vybere toho nejlepšího.

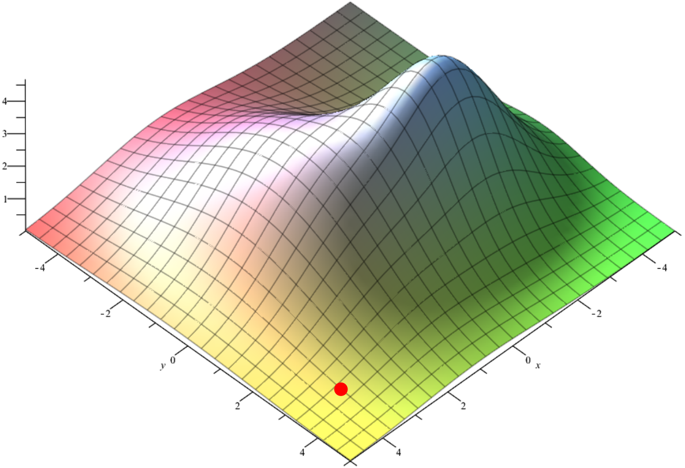
Algorithm 5 Steepest ascent Hill climbing

```
x ← random_state()
i ← 0
while ¬good_enough(x) ∧ i < max_iter do
    y ← random_neighbor(x)
    for i = 2, 3, ..., k do
        z ← random_neighbor(x)
        if f(z) > f(y) then
            y ← z
        end if
    end for
    if f(y) > f(x) then
        x ← y
    end if
    i ← i + 1
end while
return x
```

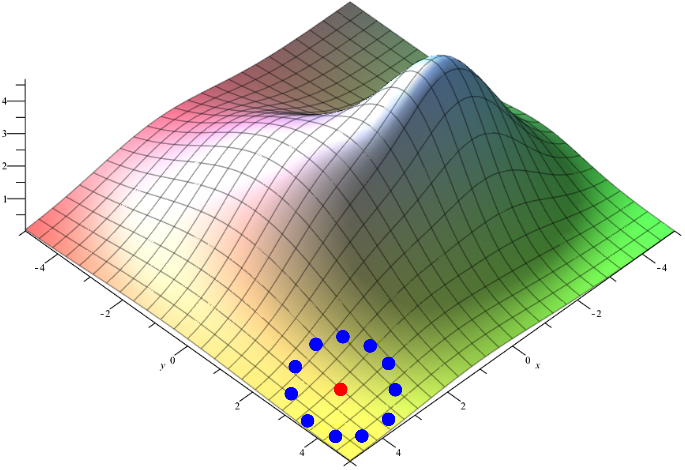
Steepest ascent Hill climbing: Ilustrace



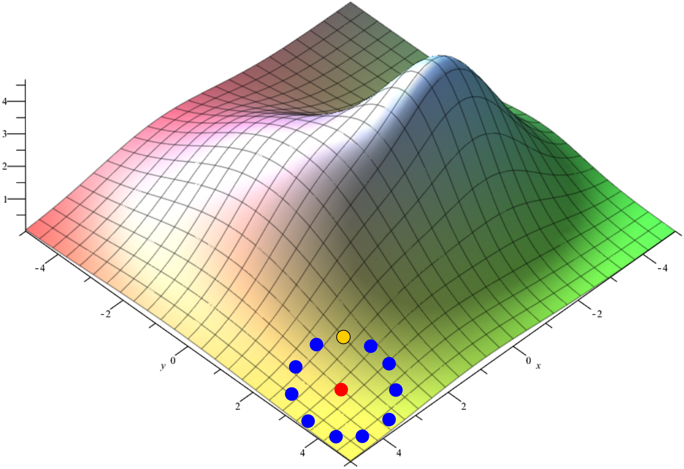
Steepest ascent Hill climbing: Ilustrace



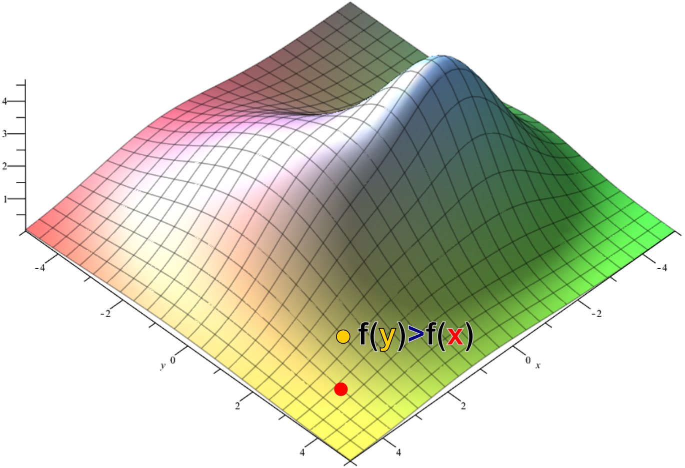
Steepest ascent Hill climbing: Ilustrace



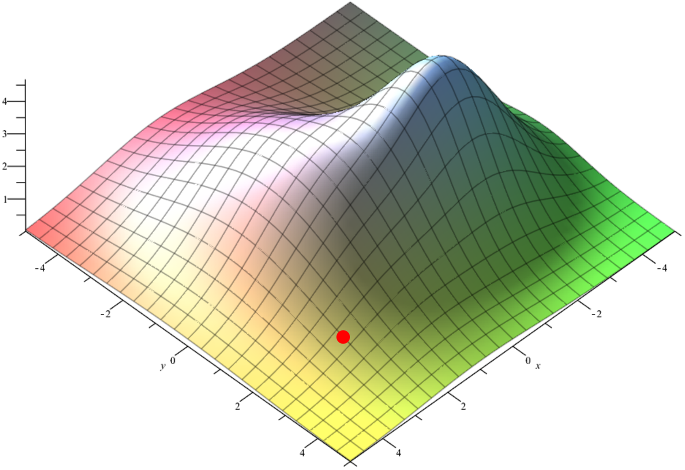
Steepest ascent Hill climbing: Ilustrace



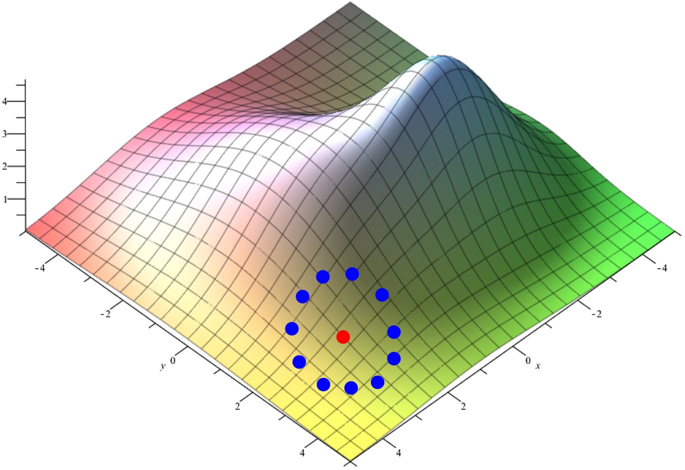
Steepest ascent Hill climbing: Ilustrace



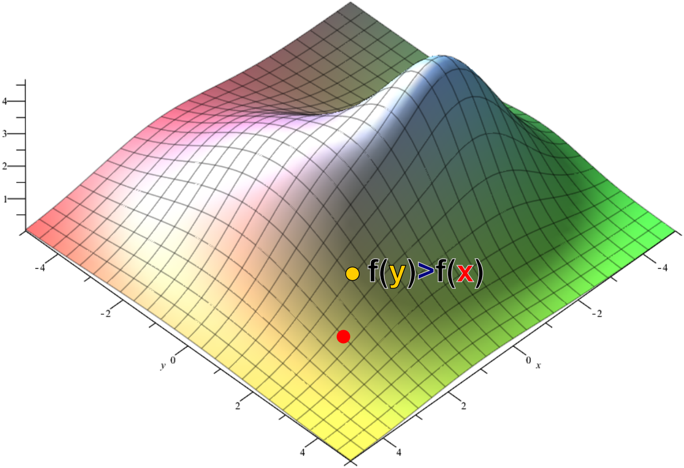
Steepest ascent Hill climbing: Ilustrace



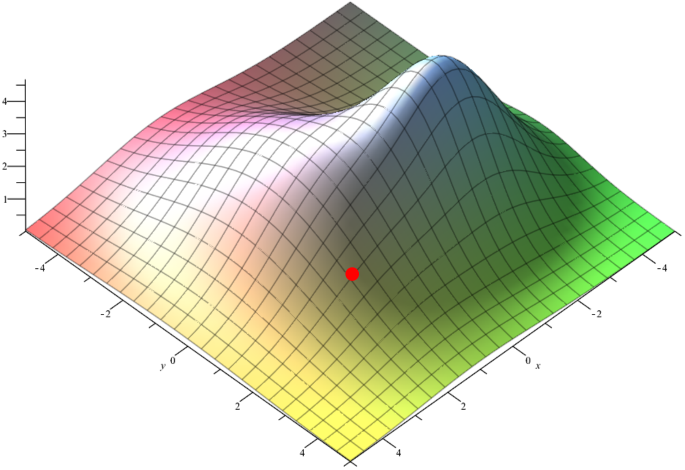
Steepest ascent Hill climbing: Ilustrace



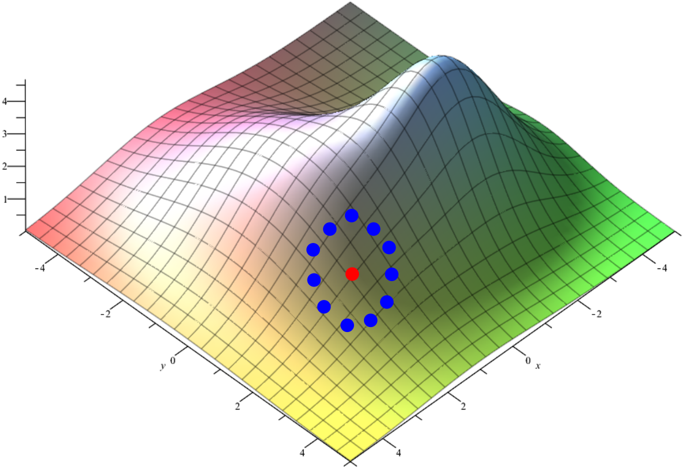
Steepest ascent Hill climbing: Ilustrace



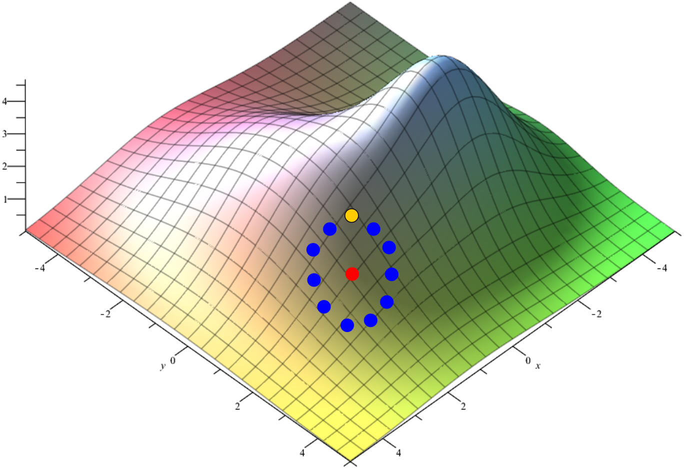
Steepest ascent Hill climbing: Ilustrace



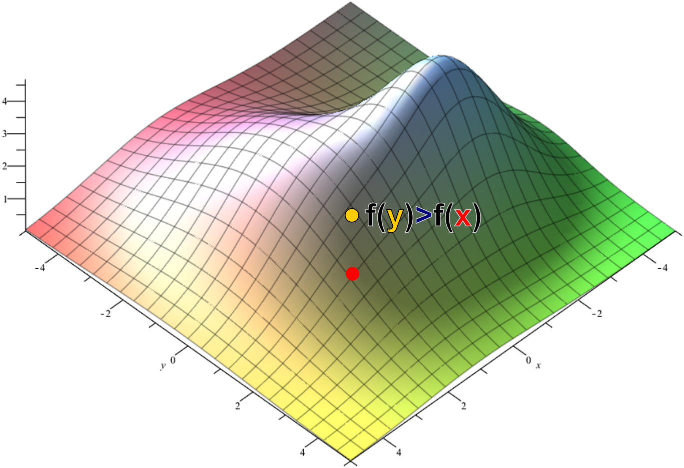
Steepest ascent Hill climbing: Ilustrace



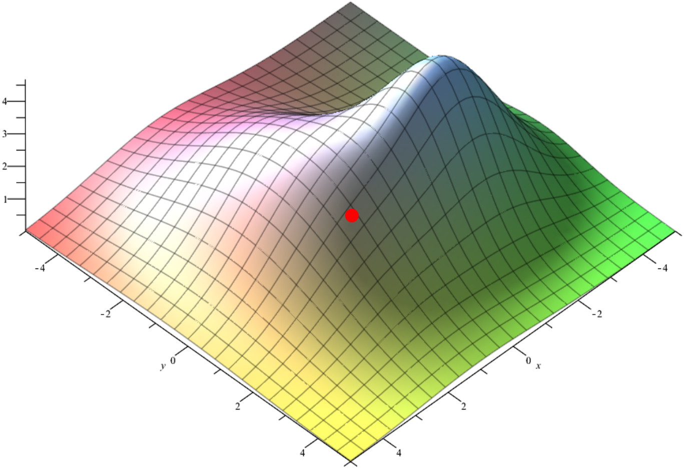
Steepest ascent Hill climbing: Ilustrace



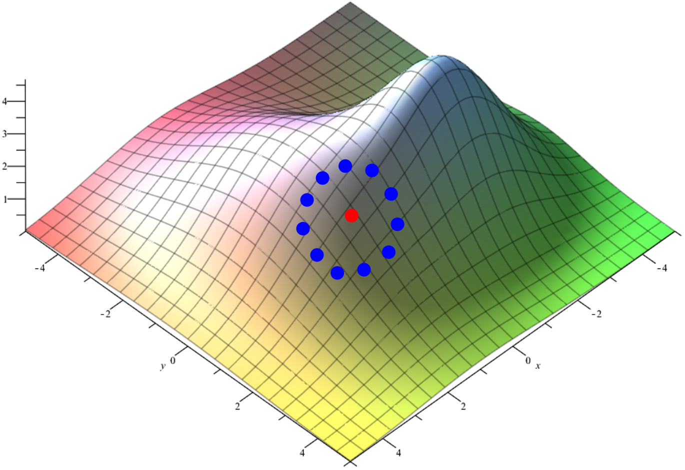
Steepest ascent Hill climbing: Ilustrace



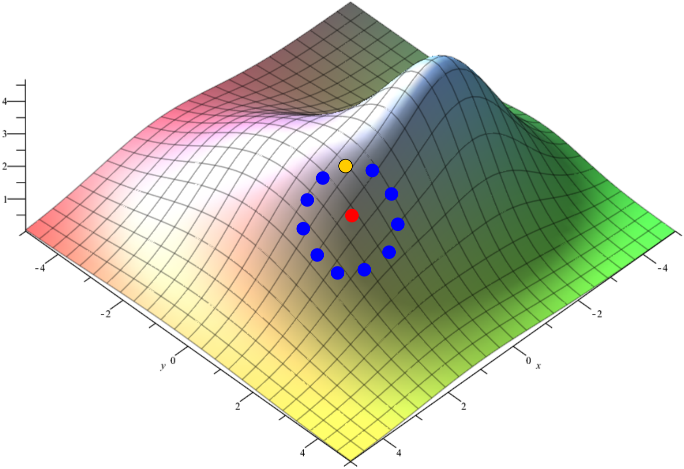
Steepest ascent Hill climbing: Ilustrace



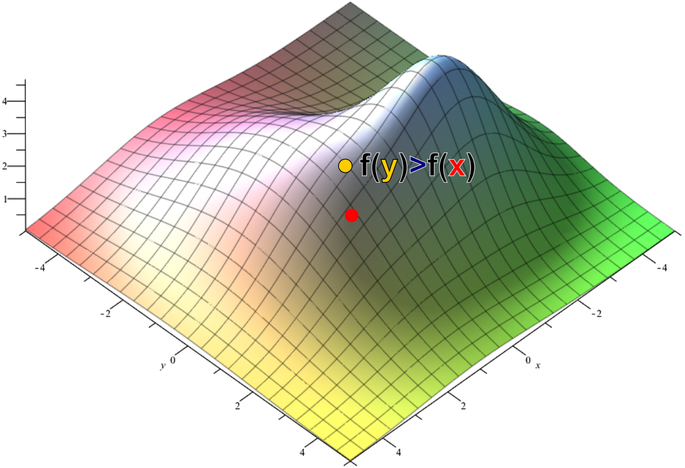
Steepest ascent Hill climbing: Ilustrace



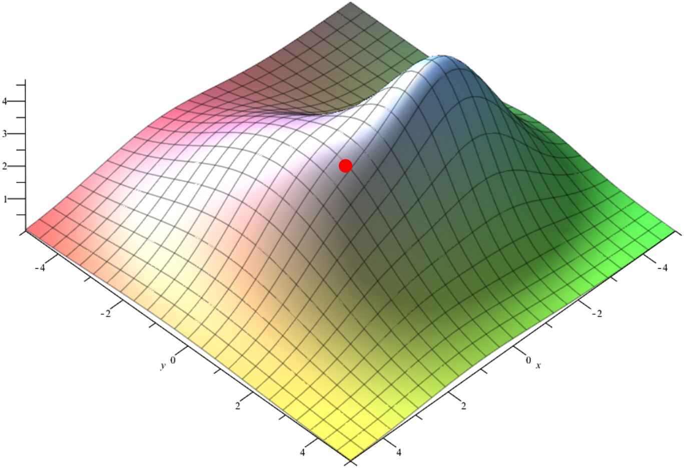
Steepest ascent Hill climbing: Ilustrace



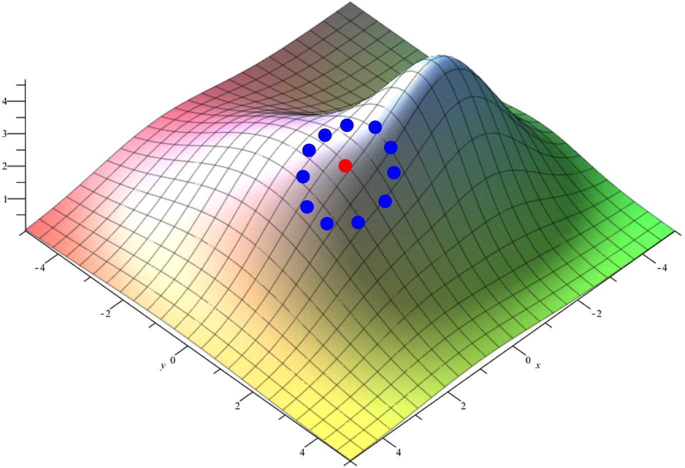
Steepest ascent Hill climbing: Ilustrace



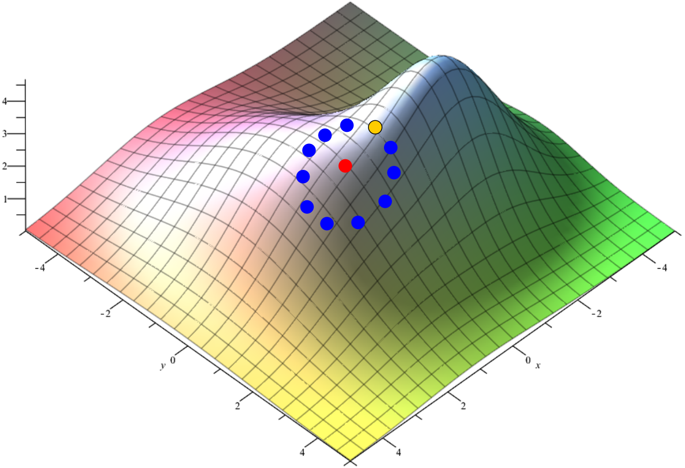
Steepest ascent Hill climbing: Ilustrace



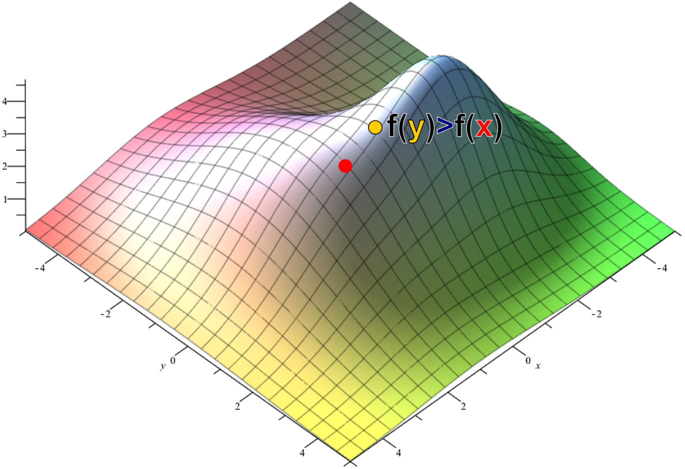
Steepest ascent Hill climbing: Ilustrace



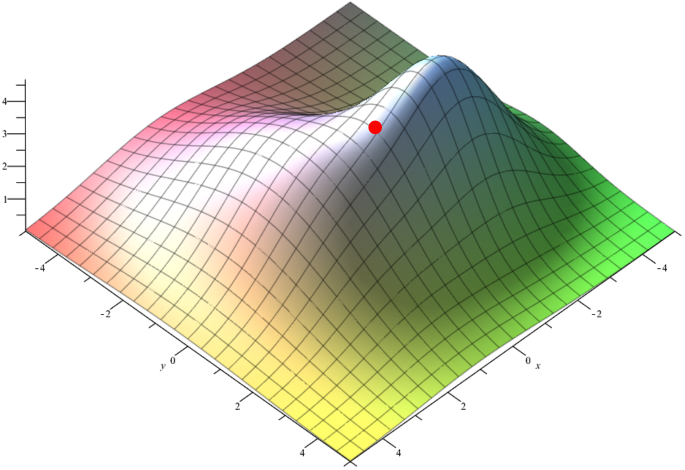
Steepest ascent Hill climbing: Ilustrace



Steepest ascent Hill climbing: Ilustrace



Steepest ascent Hill climbing: Ilustrace



Pozor, reklama!

Aplikovaný hill-climbing je možno absolvovat na turistických tělovýchovných kurzech ÚTVS ČVUT!


Relevantní jsou zejména kurzy:

- **Adršpach** (kód ADR01)
  - `http://www.utvs.cvut.cz/letni-kurzy/destinace/adrspach.html`


- **Jeseníky** (kód JES01)
  - `http://www.utvs.cvut.cz/letni-kurzy/destinace/jeseniky.html`

Naplněnost kurzů stále roste!

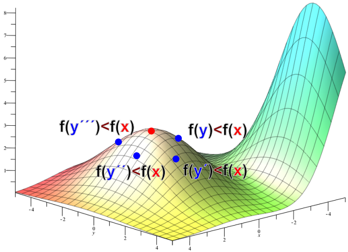
Pokyny pro zápis:

`http://www.utvs.cvut.cz/letni-kurzy/pokyny-pro-zapis.html`

Hill climbing: Problémy

Dva největší problémy hill climbingu (a iterativní optimalizace vůbec):

- **Lokální extrémy**
  - Ke zlepšení řešení je potřeba učinit dočasně zhoršující kroky

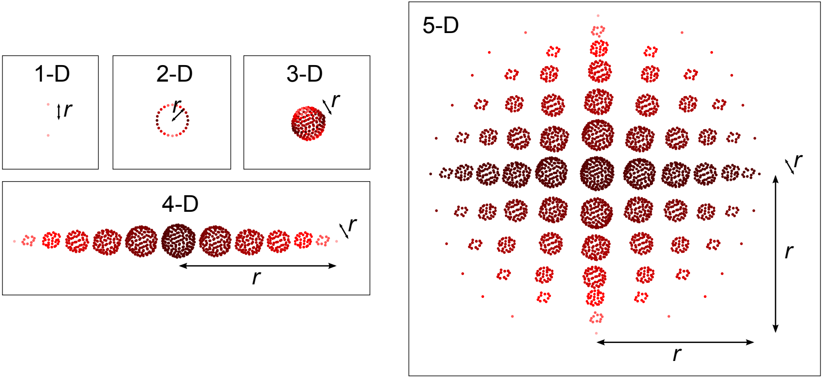


- **Prokletí dimenzionality**
  - Pozitivní vlastnosti prostoru nízké dimenze (pro nás familiární 3D) přestávají v prostorech vyšších dimenzí platit,
  - Objem prostoru roste exponenciálně s jeho dimenzí a není tedy možné s dostatečnou hustotou navzorkovat okolí bodu

Hill climbing: Problémy

Prokletí dimenzionality

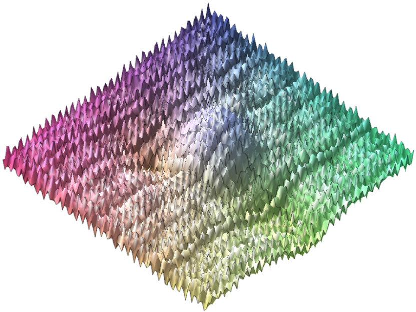
Chceme-li rovnoměrně vzorkovat body v okolí vektoru z $\mathbb{R}^n$, potřebujeme počet vzorků rostoucí exponenciálně s n.



Simulované žíhání

též Simulované popouštění, Simulované ochlazování, angl. *Simulated annealing*

- Většina kriteriálních funkcí je v praxi zašuměná a s ohromným množstvím lokálních optim



Dilema: Jak velké volit okolí?

- Volíme-li moc malé okolí, uvízneme v lokálním optimu (nepřekonáme údolí)
- Volíme-li moc velké okolí, pravděpodobně kvůli velkým krokům budeme míjet lokální optima

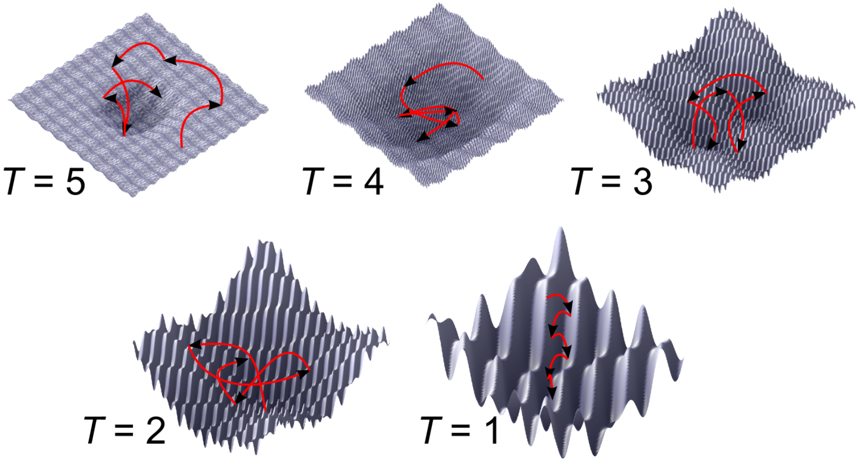
Řešení? **Simulované žíhání**!

Simulované žíhání

- Žíhání?
  - Proces tepelné úpravy oceli za účelem zlepšení jejích vlastností (zejm. odstranění vnitřního pnutí),
  - Při vysoké teplotě jsou se atomy uvolní z krystalické mřížky a je-li materiál následně pozvolna ochlazován, jsou schopny se do mřížky pravidelně uspořádat
- V kontextu optimalizace:
  - Klasický Hill climbing rozšíříme o řídící parametr t, teplotu,
  - Pokud $f(\mathbf{y}) > f(\mathbf{x})$, standardně přejdeme k novému řešení,
  - Pokud $f(\mathbf{y}) \leq f(\mathbf{x})$, potenciálně přejdeme k tomuto zhoršujícímu řešení, a to s pravděpodobností závislou na
    - míře zhoršení $\left(f(\mathbf{y}) - f(\mathbf{x})\right)$: čím větší zhoršení, tím nižší pravděpodobnost,
    - aktuální teplotě t: čím vyšší teplota, tím vyšší pravděpodobnost,

Simulované žíhání: Obrázek

Funkci na obrázku **minimalizujeme** (lépe odpovídá žíhání, kde atomy usedají do mřížky).



Simulované žíhání: Pseudokód

Algorithm 6 Simulated annealing

```
x ← random_state()
t ← high_number
while t ≥ 0 do
    y ← random_neighbor(x)
    if f(y) > f(x) then
        x ← y
    else if P(f(x), f(y), t) ≥ random(⟨0, 1⟩) then
        x ← y
    end if
    t ← decrease()
end while
return x
```

Typická pravděpodobnostní funkce:

$$\mathrm{P}(f_{curr}, f_{new}, t) = \mathrm{e}^{\frac{f_{new} - f_{curr}}{t}}$$

Tabu prohledávání

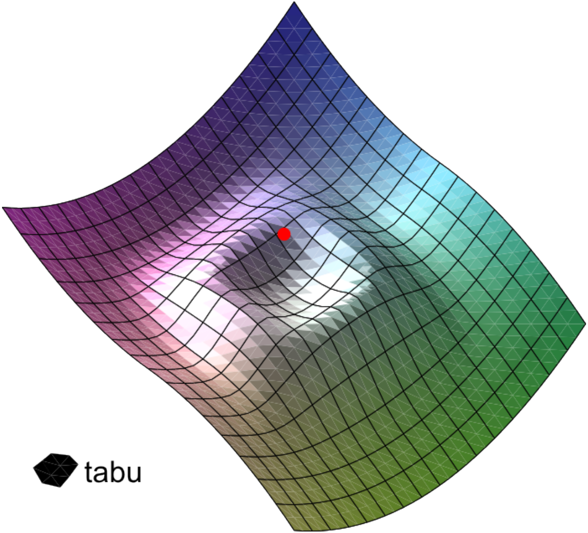
Princip algoritmu

- Snaha zabránit **oscilaci** a přinutit optimalizační algoritmus vymanit se z lokálního optima,
- Zavádí tzv. **tabu list**, který popisuje části prostoru, kam se kandidující řešení nesmí vrátit,
- Pokud algoritmus vyšplhal do kopce, je nucen „dobytý vrchol" opustit a zahájit vynucený sestup,

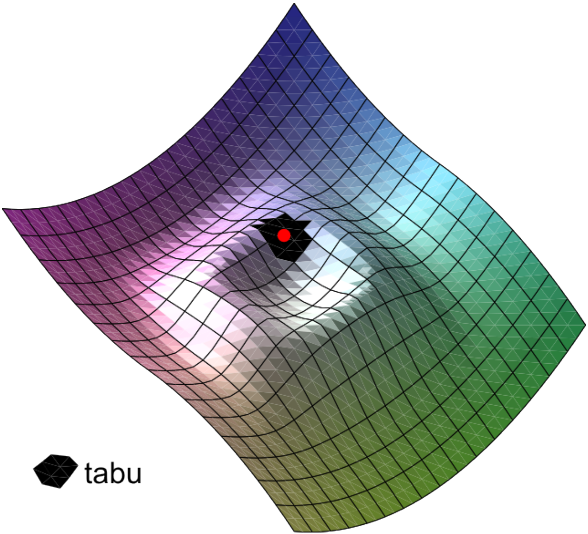
Podoba *tabu listu*?

- Pro různé problémy různá,
- většinou vycházíme z podobnosti (eventuálně metriky) na množině stavů:
  - Euklidovská nebo cosinová vzdálenost pro vektory z $\mathbb{R}^n$,
  - Hammingova vzdálenost pro vektory z $\{0, 1\}^n$,
  - Strukturální podobnost pro grafy/stromy. . .

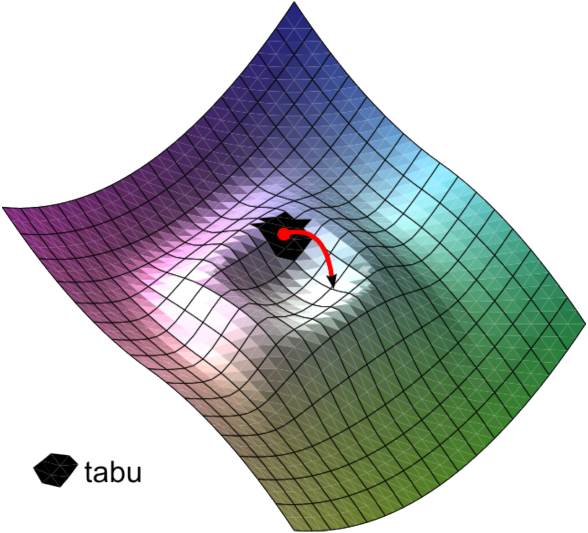
Tabu prohledávání: Ilustrace (minimalizace *f*)



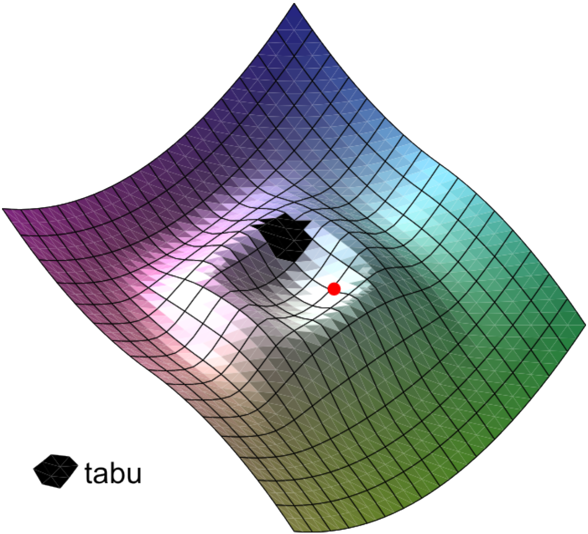
Tabu prohledávání: Ilustrace (minimalizace f)



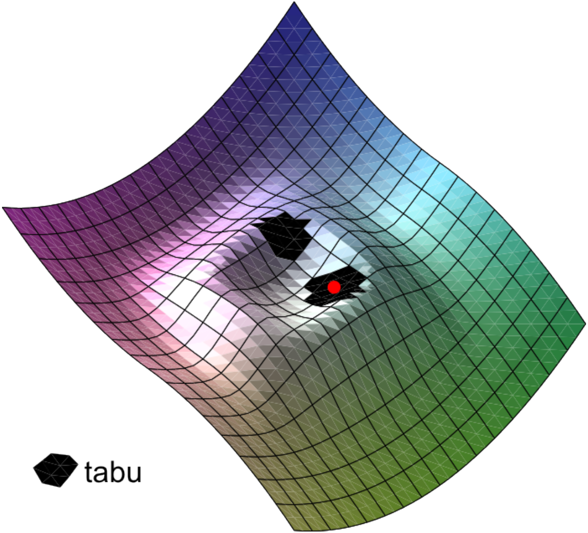
Tabu prohledávání: Ilustrace (minimalizace f)



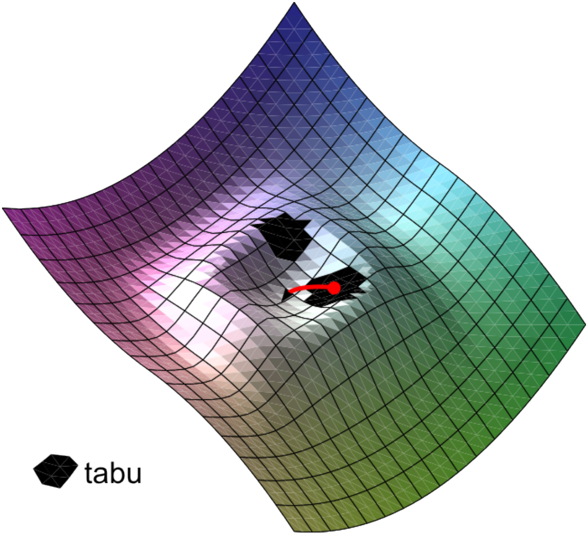
Tabu prohledávání: Ilustrace (minimalizace f)



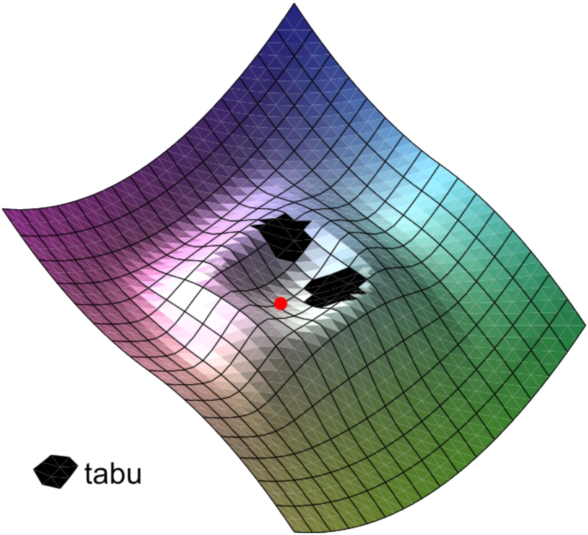
Tabu prohledávání: Ilustrace (minimalizace f)



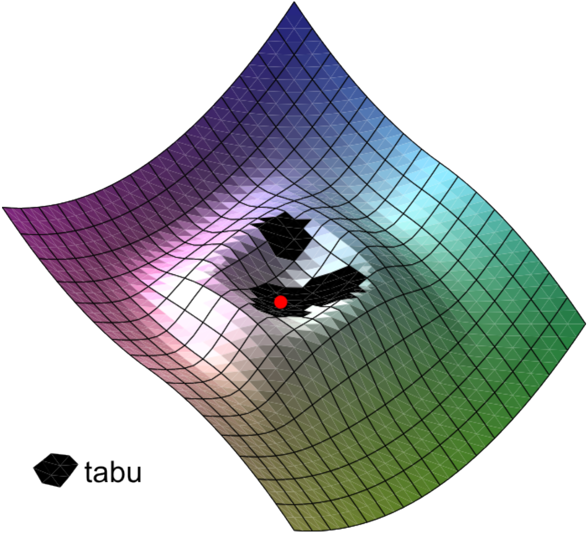
Tabu prohledávání: Ilustrace (minimalizace f)



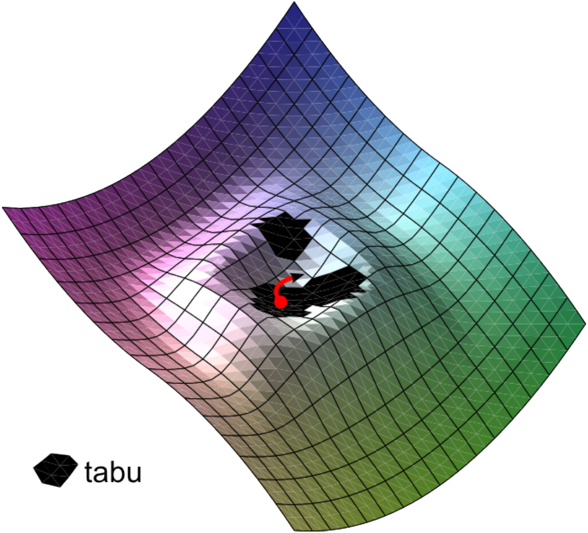
Tabu prohledávání: Ilustrace (minimalizace f)



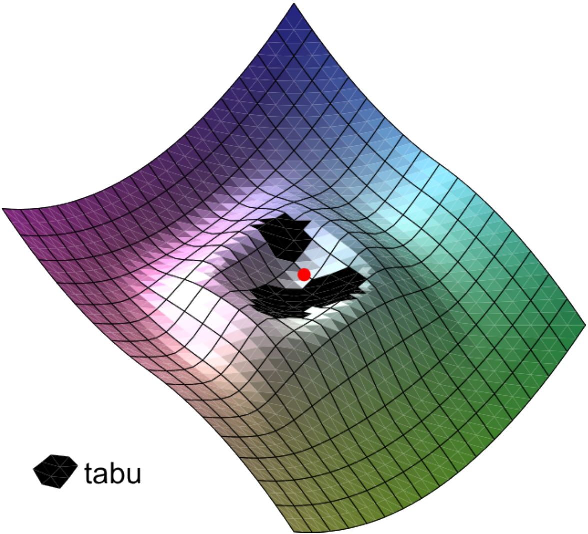
Tabu prohledávání: Ilustrace (minimalizace f)



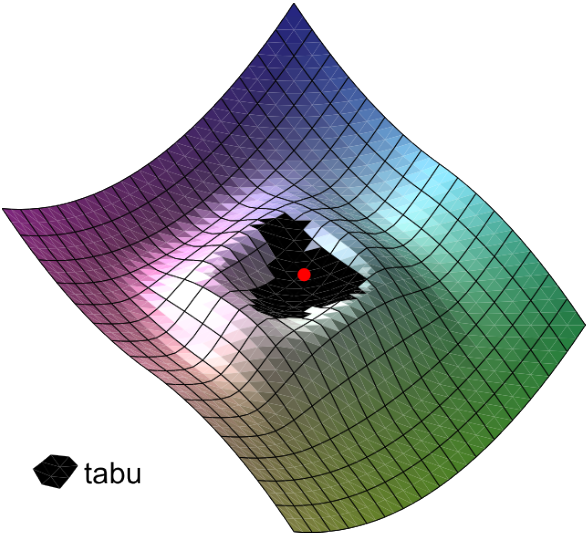
Tabu prohledávání: Ilustrace (minimalizace *f*)



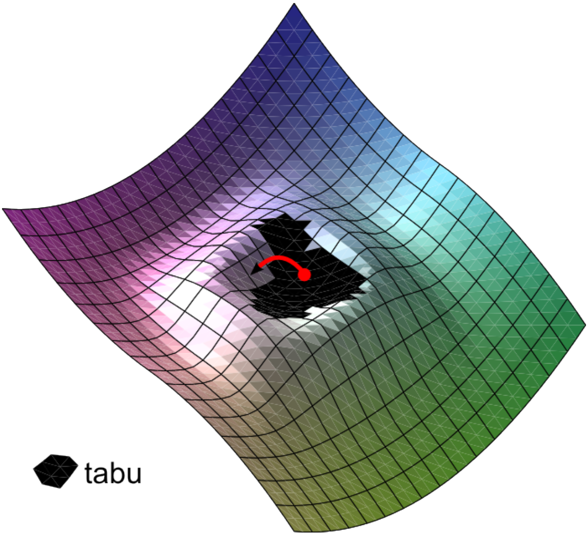
Tabu prohledávání: Ilustrace (minimalizace f)



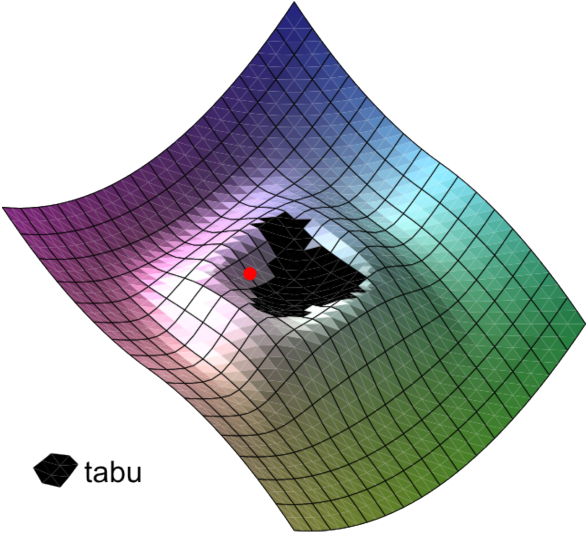
Tabu prohledávání: Ilustrace (minimalizace f)



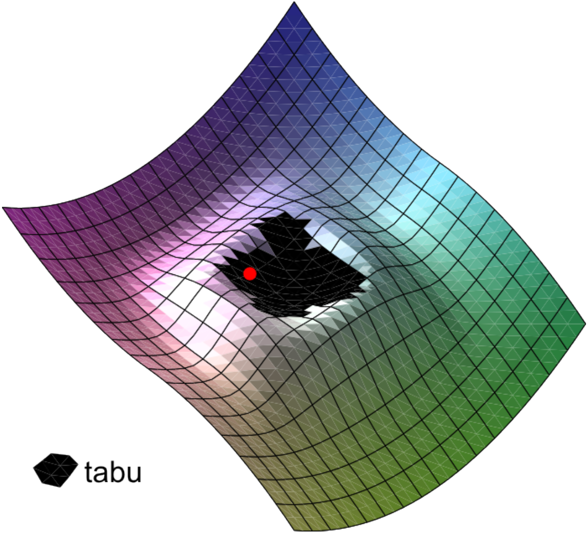
Tabu prohledávání: Ilustrace (minimalizace f)



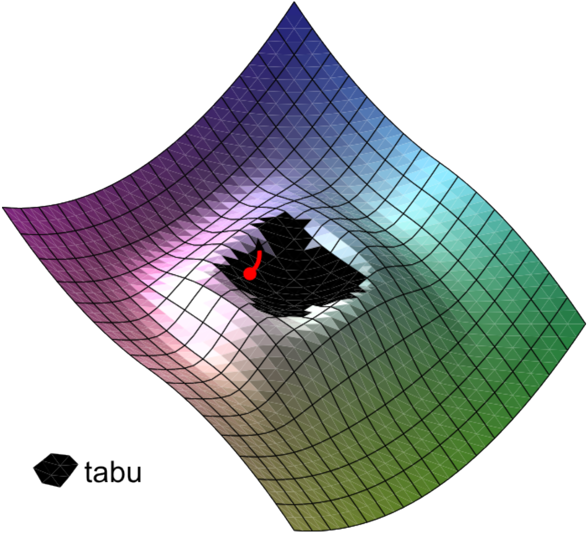
Tabu prohledávání: Ilustrace (minimalizace f)



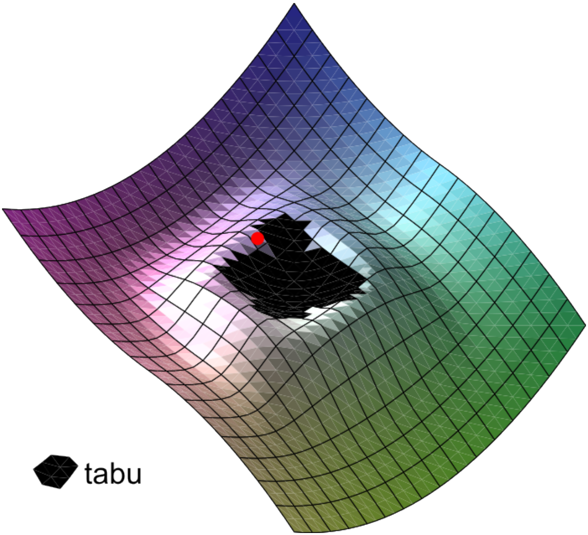
Tabu prohledávání: Ilustrace (minimalizace f)



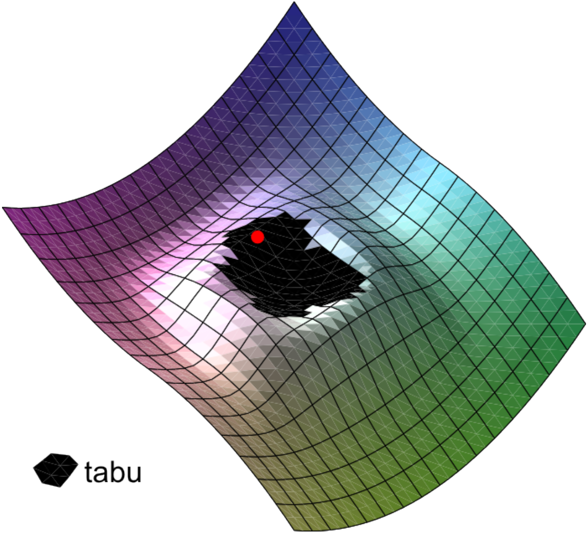
Tabu prohledávání: Ilustrace (minimalizace f)



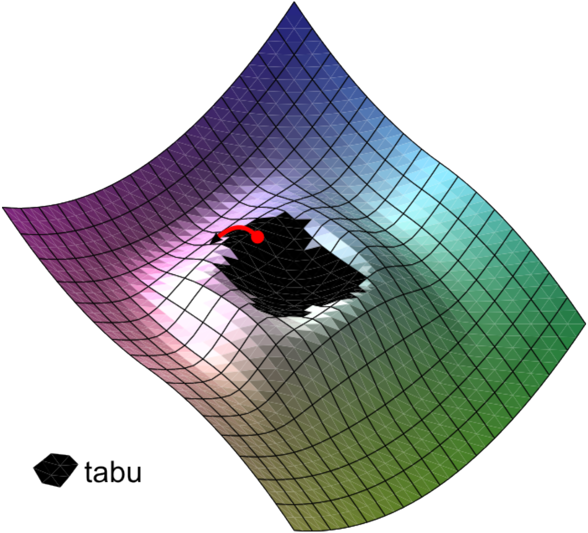
Tabu prohledávání: Ilustrace (minimalizace f)



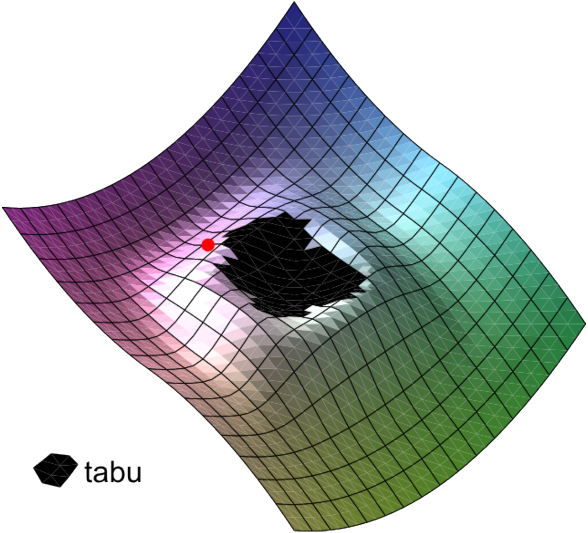
Tabu prohledávání: Ilustrace (minimalizace f)



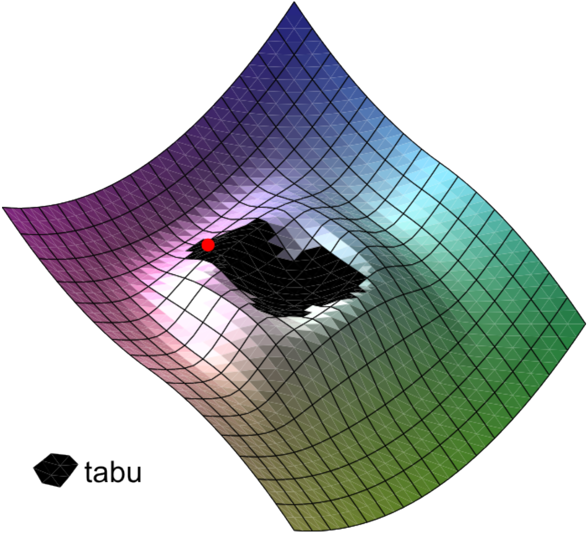
Tabu prohledávání: Ilustrace (minimalizace f)



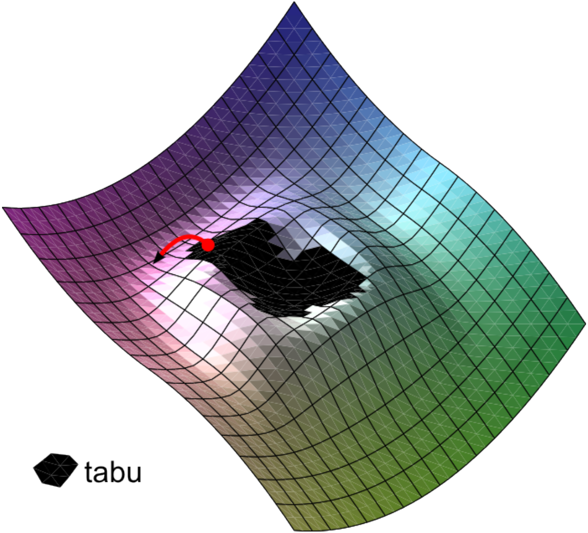
Tabu prohledávání: Ilustrace (minimalizace f)



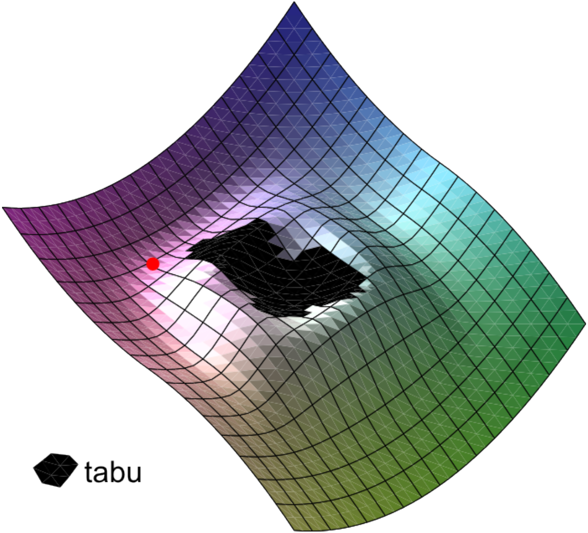
Tabu prohledávání: Ilustrace (minimalizace f)



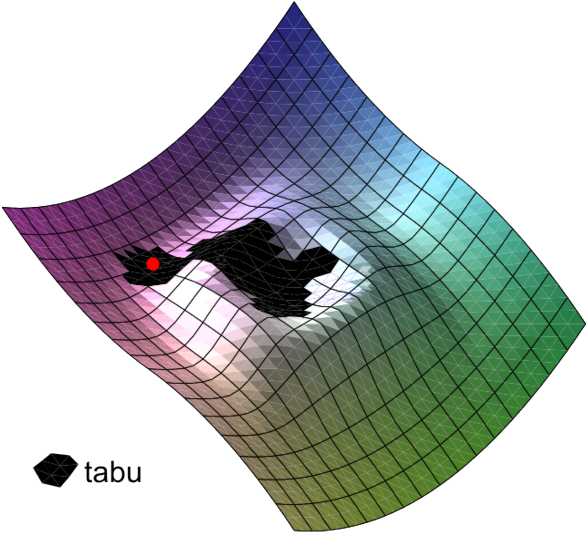
Tabu prohledávání: Ilustrace (minimalizace f)



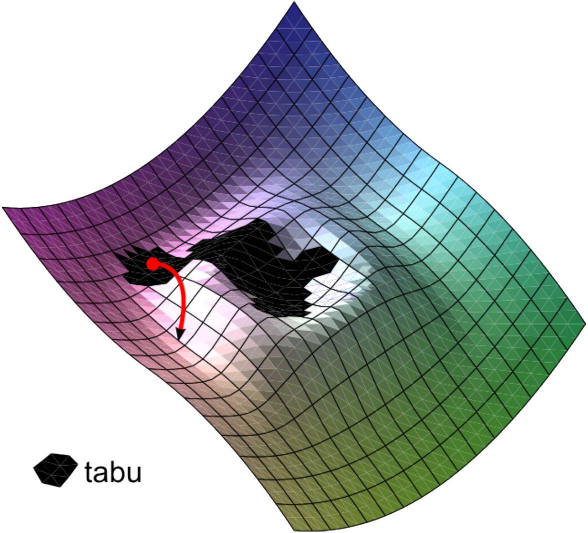
Tabu prohledávání: Ilustrace (minimalizace f)



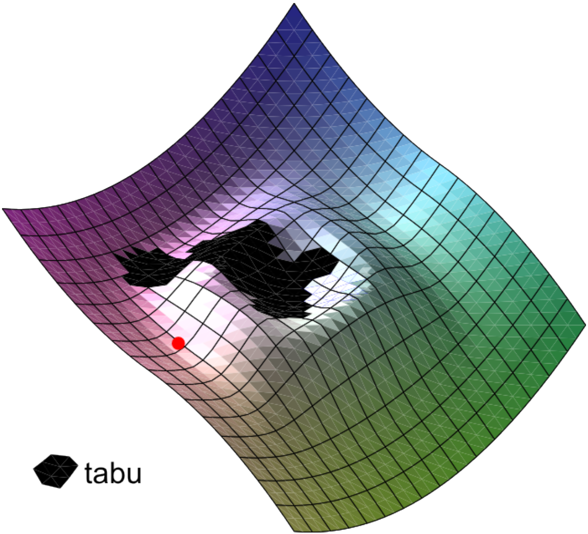
Tabu prohledávání: Ilustrace (minimalizace f)



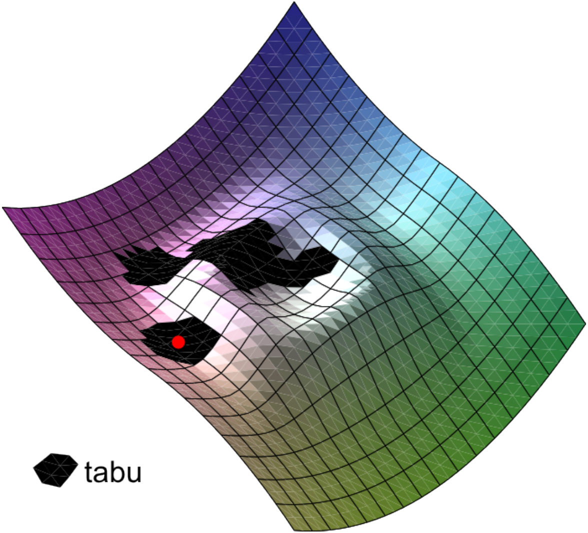
Tabu prohledávání: Ilustrace (minimalizace f)



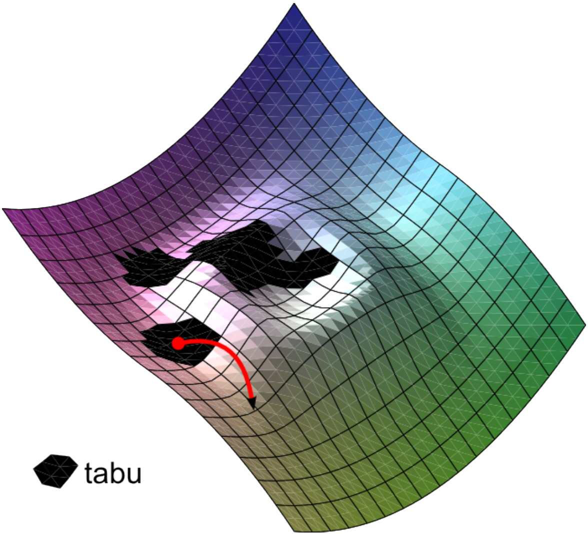
Tabu prohledávání: Ilustrace (minimalizace f)



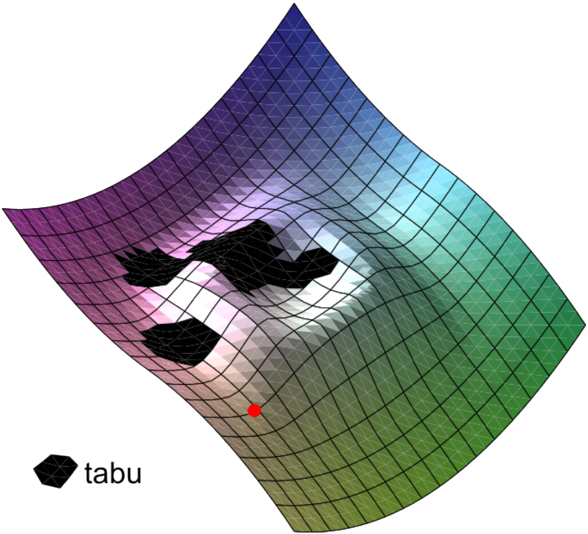
Tabu prohledávání: Ilustrace (minimalizace f)



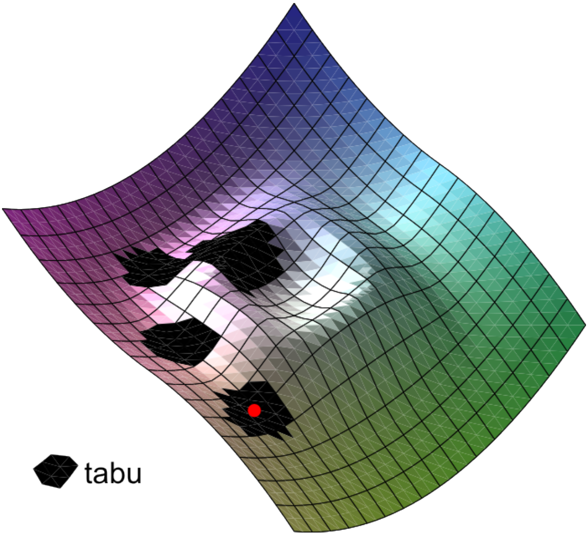
Tabu prohledávání: Ilustrace (minimalizace f)



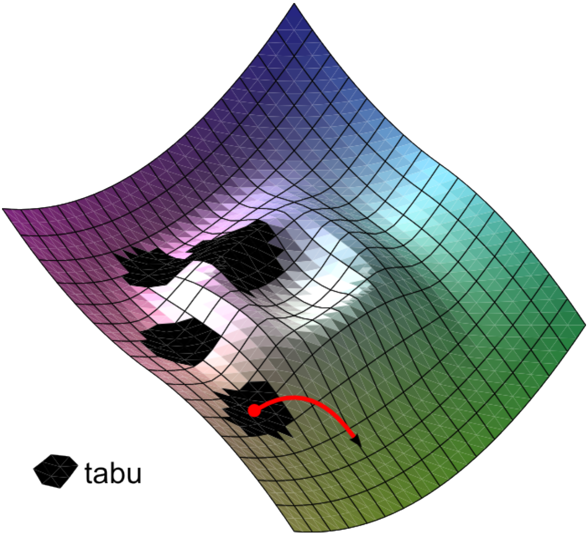
Tabu prohledávání: Ilustrace (minimalizace f)



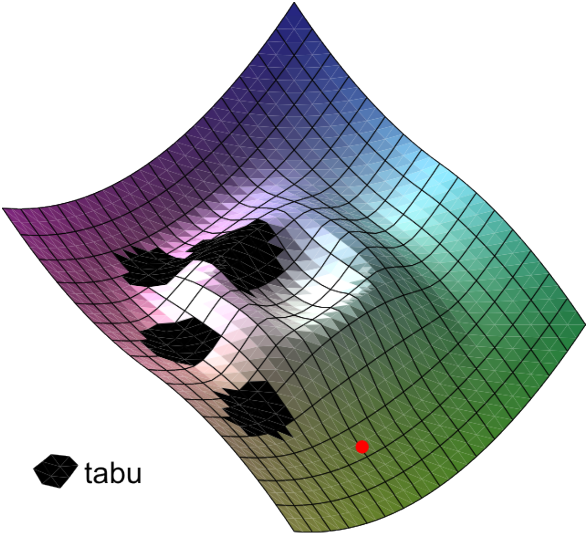
Tabu prohledávání: Ilustrace (minimalizace f)



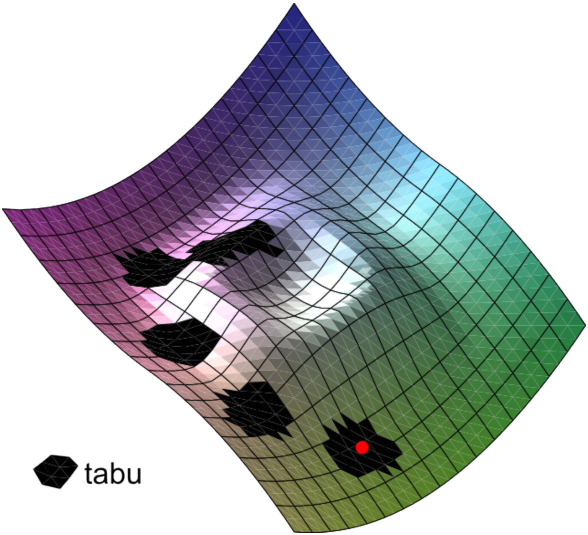
Tabu prohledávání: Ilustrace (minimalizace f)



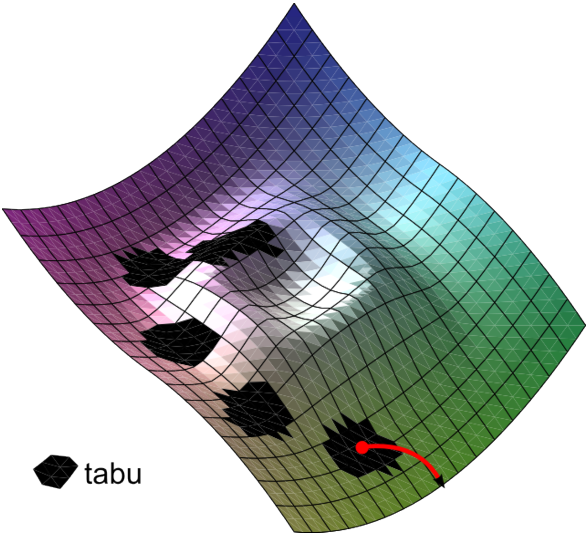
Tabu prohledávání: Ilustrace (minimalizace f)



Tabu prohledávání: Ilustrace (minimalizace f)



Tabu prohledávání: Ilustrace (minimalizace f)



Populační metody

Všechny dosud uvedené algoritmy pracují s **jediným** kandidujícím řešením

- Hill climbing, Simulované žíhání, Tabu search...

Co dělat v případě, že je optimalizační problém příliš těžký?

- Algoritmy lze spouštět opakovaně, případně paralelně (např. 1000$\times$)

Populační metody jdou ještě dále:

- Pracují s několika kandidujícími řešeními (populací), ovšem zavádí mezi ně **interakci**
- Kandidáti se mohou vzájemně ovlivňovat a tvořit finální řešení společným úsilím

Populační metody

Algoritmy jsou často inspirovány biologicky:

- Evoluční algoritmy
  - kandidující řešení mezi sebou soupeří o přežití,
  - silnější řešení jsou reprodukována a křížena,
  - téma příští přednášky :-)
- Particle Swarm Optimization (PSO)
  - řešení „létají“ prostorem a korigují svou dráhu letu vzhledem k nejlepším z nich,
- . . . a mnoho dalších:
  - umělé mravenčí kolonie (Ant Colony Optimization, ACO),
  - inteligentní dešťové kapky (Intelligent Water Drops, IWD),
  - umělé imunitní systémy (Artificial Immune Systems, AIS),
  - umělé včelí kolonie (Bee Colony Optimization, BCO), . . .