Problém batohu - 4. úloha

MI-PAA, zimný semester 2014

Martin Klepáč

November 29, 2014

1 Definícia problému

Problém batohu (knapsack problem) je jeden z NP-ťažkých problémov. Pokiaľ máme k dispozícii množinu predmetov, pričom každý predmet má svoju váhu, cenu, a maximálnu nosnosť batohu, potom je naším cieľom vybrať predmety do batohu tak, aby sme maximalizovali ich cenu a pritom celková hmotnosť predmetov nepresiahla maximálnu povolenú nosnosť batohu.

2 Formát vstupu

Formálne, vstup vyjadríme pomocou

- n = počet predmetov
- M = maximálna nosnosť batohu
- *vektor* w = hmotnosť jednotlivých predmetov
- *vektor* c = hodnota (cena) jednotlivých predmetov

Výstupom algoritmu je vektor x, ktorý udáva prítomnosť/neprítomnosť daného predmetu vo výbere. Ďalej požadujeme informáciu o celkovej hmotnosti predmetov v batohu a ich súhrnú hodnotu pre najlepší výber.

3 Úvod

V poslednej úlohe venovanej problému batohu je cieľom otestovať si nasadenie algoritmu simulovaného ochladzovania respektíve genetického algoritmu na tento problém. Okrem samotnej implementácie je potrebné experimentálne zhodnotiť výkonnosť algoritmu pri zmene parametrov.

4 Simulované ochladzovanie

Pre túto úlohu som si vybral algoritmus simulovaného ochladzovania, ktorý mi pripadá jednoduchší na pochopenie. Simulované ochladzovanie predstavuje lokálnu metódu prehľadávania stavového priestoru. Metóda umožňuje s určitou pravdepodobnosťou prijať aj stav horší od aktuálneho stavu, ale táto pravdepodobnosť klesá s postupným ochladzovaním, čo v konečnom dôsledku zaisťuje konvergenciu algoritmu. Pseudokód metódy vyzerá nasledovne:

```
function simulatedAnnealing ()
{
  current_temp = INITIAL_TEMP;
  state = INITIAL_STATE;
  while (current_temp > LOWER_BOUND_TEMP)
  {
    steps = NO_OF_STEPS;
    while (steps > 0)
    {
      state = newState (state, current_temp);
      if (state > bestState)
      {
        bestState = state;
      }
      steps--;
    }
    current_temp = cool (current_temp)
  }
}
```

Cool je funkcia, ktorá zníži aktuálnu teplotu aplikovaním koeficientu ochladzovania *COOLING_FACTOR*. *NewState* je funkcia, ktorá vracia nejaký nový stav, pričom pravdepodobnosť, že funkcia vráti horší stav ako aktuálny klesá s poklesom aktuálnej teploty *current_temp*. Jej implementácia je zhodná so slidom číslo 13 prednášky číslo 8 - Simulované ochlazování. Výstupom algoritmu je *bestState* - najlepší nájdený výber.

5 Výsledky

Merania prebiehali na 64-bitovom Linuxovom hostiteľovi použitom v predchádzajúcich úlohách. Pre potreby vyladenia algoritmu som experimentálne zisťoval hodnoty počiatočnej a koncovej teploty (v pseudokóde vyššie označené ako *INITIAL_TEMP* resp. *LOWER_BOUND_TEMP*), ďalej koeficient ochladzovania *COOLING_FACTOR* a počet iterácií vnútorného cyklu - *NO_OF_STEPS*.

Merania prebiehali iteratívne, čo znamená, že najskôr som zmeral závislosť doby behu algoritmu a priemernej/maximálnej relatívnej chyby v závislosti od variabilného koeficientu ochladzovania (ostatné premenné zostali zafixované s určitými hodnotami).

Následne som z výsledkov odhadol optimálny koeficient ochladzovania a začal študovať závislosť od počiatočnej teploty. K tomu bolo potrebné zafixovať ostatné premenné na určitých hodnotách s tým, že optimálny koeficient ochladzovania som už mal experimentálne zistený. Takto som vždy jednou sadou merania zistil optimálnu hodnotu ďalšieho parametru. Konečne, po zistení optimálnych hodnôt všetkých parametrov som všetky experimenty vykonal ešte raz - jednu premennú som menil a všetky ostatné boli zafixované s použítím zisteného optima. Výsledky pre veľkosť inštancie *n=40* uvádzam v tabuľkách 1 až 4 resp. v grafoch 1 až 4. Graf 5 zobrazuje závislosť ceny riešenia od aktuálneho čísla iterácie pre jednú konkrétnu inštanciu zo vzorky inštancií pre n=40.

Koeficient ochladzovania	**Doba behu [s]**	**Priem. rel. chyba [%]**	**Max. rel. chyba [%]**
c=0.800	0.012 644	0.111	2.3372
c=0.825	0.014 974	0.088	2.3372
c=0.850	0.018 121	0.058	2.3372
c=0.875	0.021 101	0.048	2.3372
c=0.900	0.026 915	0.058	2.3372
c=0.925	0.037 111	0.054	2.3372
c=0.950	0.055 780	0.047	2.3372
c=0.975	0.110 550	0.052	2.3372

Table 1: Závislosť trvania doby behu a relatívnej chyby simulovaného ochladzovania od koeficientu ochladzovania

Počiatočná teplota	**Doba behu [s]**	**Priem. rel. chyba [%]**	**Max. rel. chyba [%]**
t=50	0.009 352	0.103	2.3372
t=100	0.011 937	0.063	2.3372
t=200	0.014 823	0.068	2.3372
t=400	0.017 581	0.058	2.3372
t=700	0.020 288	0.072	2.3372
t=1 000	0.021 640	0.084	2.3372
t=1 500	0.024 060	0.082	2.3372
t=2 000	0.023 636	0.087	2.3372
t=3 000	0.025 271	0.072	2.3372
t=5 000	0.027 588	0.097	2.3372

Table 2: Závislosť trvania doby behu a relatívnej chyby simulovaného ochladzovania od počiatočnej teploty

Konečná teplota	Doba behu [s]	Priem. rel. chyba [%]	Max. rel. chyba [%]
t=0.5	0.025 798	0.062	2.3372
t=1	0.023 568	0.088	2.3372
t=2	0.022 017	0.078	2.3372
t=5	0.017 258	0.058	2.3372
t=10	0.014 903	0.105	4.8414
t=20	0.012 658	0.246	9.8497

Table 3: Závislosť trvania doby behu a relatívnej chyby simulovaného ochladzovania od konečnej teploty

Počet iterácií	Doba behu [s]	Priem. rel. chyba [%]	Max. rel. chyba [%]
no=100	0.000 197	2.501	5.4936
no=1 000	0.001 773	0.535	3.7279
no=10 000	0.017 543	0.058	2.3372
no=100 000	0.175 100	0.046	2.3372
no=1 000 000	1.676 700	0.046	2.3372

Table 4: Závislosť trvania doby behu a relatívnej chyby simulovaného ochladzovania od počtu iterácií

6 Zhodnotenie

Pomocou metodiky v sekcii Výsledky som dospel k nasledovným optimálnym hodnotám parametrov:

- INITIAL_TEMP = 400
- LOWER_BOUND_TEMP = 5
- COOLING_FACTOR = 0.85
- NO_OF_STEPS = 10 000

Obecne, otázku určenia optimálnych hodnôt jednotlivých parametrov môžeme preformulovať nasledovne - v prípade, ak algoritmus simulovaného ochladzovania necháme bežať dlhšie, o koľko dôjde k zmenšeniu relatívnych chýb v porovnaní s exaktnými riešeniami, akým je napríklad algoritmus dynamického programovania?

Ukázalo sa, že vysoký chladiaci pomer, nad 0.9 (=pomalé chladenie) výrazným spôsobom neznižuje relatívne chyby v porovnaní s optimálnou hodnotou 0.85.

Z hľadiska počiatočnej teploty sa ako optimálne ukázali byť hodnoty okolo 500. Pre testované inštancie o veľkosti *n=40* dosahovala najlepší pomer relatívnej chyby k dobe behu algoritmu teplota t=400, pre inú veľkosť inštancií sa ako optimálna počiatočná teplota ukázala byť hodnota 700.

Ďalej, s narastajúcou konečnou teplotou klesala na rozdiel od dvojice predchádzajúcich premenných doba behu algoritmu a zároveň dochádzalo k nárastu relatívnych

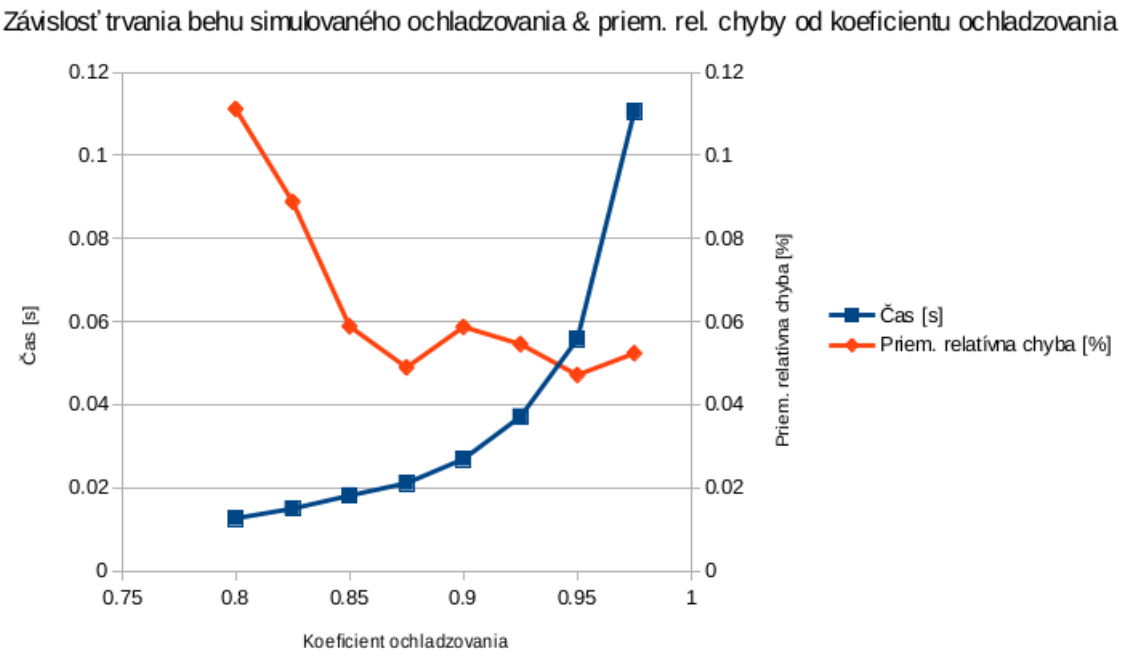

Figure 1: Závislosť trvania doby behu a relatívnej chyby simulovaného ochladzovania od koeficientu ochladzovania

chýb. Ako optimálnym riešením sa zdá byť hodnota konečnej teploty rovná 5, pri ktorej algoritmus dosahoval najmenšiu relatívnu chybu so zachovaním rozumnej doby behu.

Konečne, s narastajúcim počtom iterácií vnútorného cyklu, dochádza k rádovému rastu doby behu algoritmu pri mojich testovaných hodnotách. Rozumný pomer dosahovali hodnoty 1 000 a 10 000 iterácií. Pre otestovanie dostatočného počtu stavov som za optimálnu zvolil vyššiu hodnotu, t.j. 10 000 iterácií.

Pri týchto optimálnych hodnotách dosahuje priemerná relatívna chyba 0.058 percenta, maximálna relatívna chyba 2.33 percenta pre vzorky inštancií n=40.

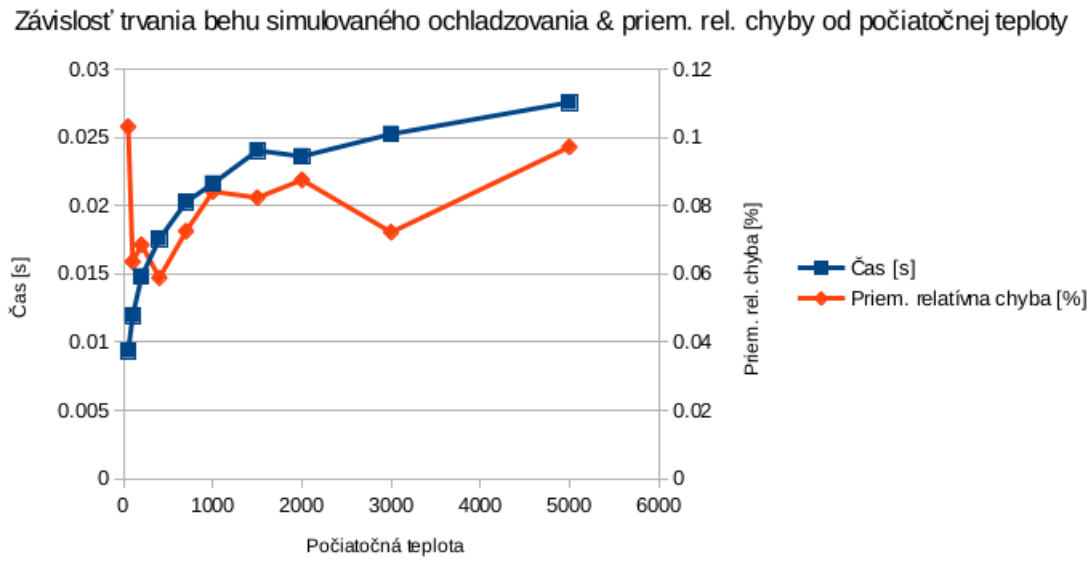

Figure 2: Závislosť trvania doby behu a relatívnej chyby simulovaného ochladzovania od počiatočnej teploty

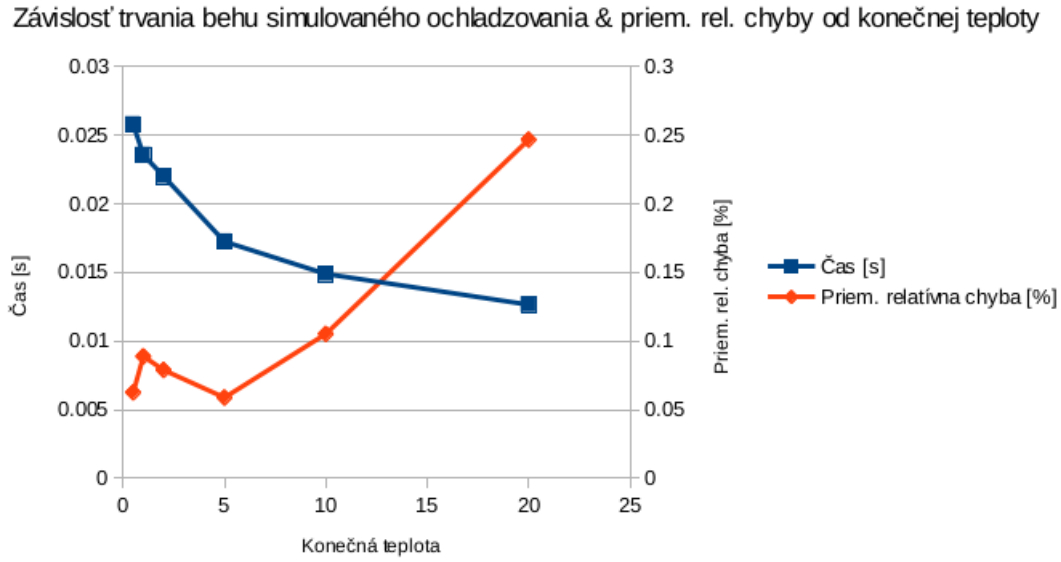

Figure 3: Závislosť trvania doby behu a relatívnej chyby simulovaného ochladzovania od konečnej teploty

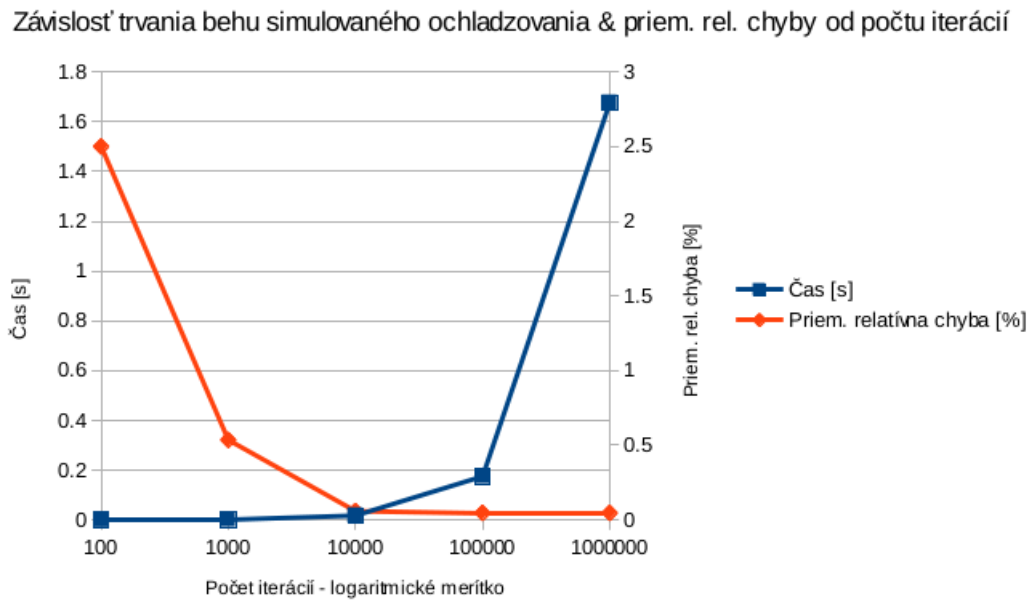

Figure 4: Závislosť trvania doby behu a relatívnej chyby simulovaného ochladzovania od počtu iterácií

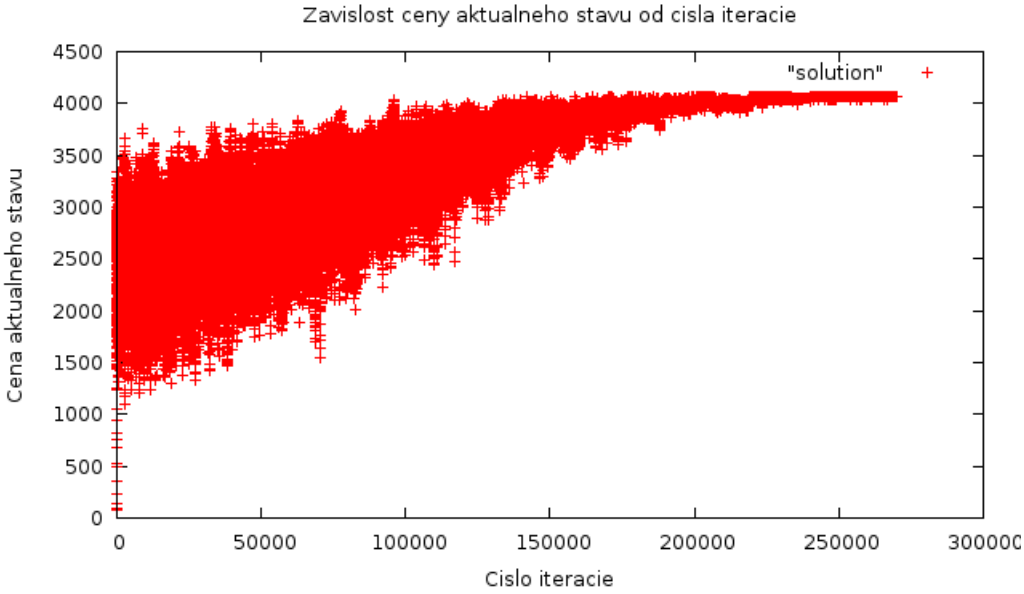

Figure 5: Závislosť ceny aktuálneho stavu od čísla iterácie