Zpráva k 1. domácímu úkolu z předmětu MI-PAA

Jan Sokol
sokolja2@fit.cvut.cz

21. října 2018

Abstrakt

Úkolem bylo nalézt řešení 0/1 problému batohu hrubou silou (tj. nalezení skutečného optimuma). Dále bylo třeba zkušebních datech pozorovat závislost výpočetního času na n (kde n je počet věcí v batohu). Druhou částí ukolu naprogramování řešení problému batohu heuristikou podle poměru cena/váha. Na těchto datech bylo třeba pozorovat závislost výpočetního času na n, a také průměrnou a maximální relativní chybu (tj. zhoršení proti exaktní metodě) v závislosti na n.

1 Parametry algoritmu, výběr jazyka

Pro svou implementaci problému batohu jsem si vybral jazyk Python. Ačkoli to je jazyk interpretovaný a nečekal jsem závratné rychlosti výpočtů, mojím výběrem byl pro to, že jsem jazyk znal a pro jakýkolik koncept je pro mne nejrychlejší.

V případě hledání řešení hroubou silou jsem těžil z materiálů v přednáškách, tak i na internetu.

Při výpočtu heuristikou jsem vybral řazení dle snižující ceny. Poté jsem objekty do batohu přidával, dokud nebylo možné přidat další věc.

2 Testovací Hardware

Všechny testy byly prováděny na cloudové linuxové instanci v AWS, běžící na Red Hat Enterprise Linux 7. Velikost instance byla: 2 Core CPU / 8 GB RAM, v názvosloví AWS **m4.large**.

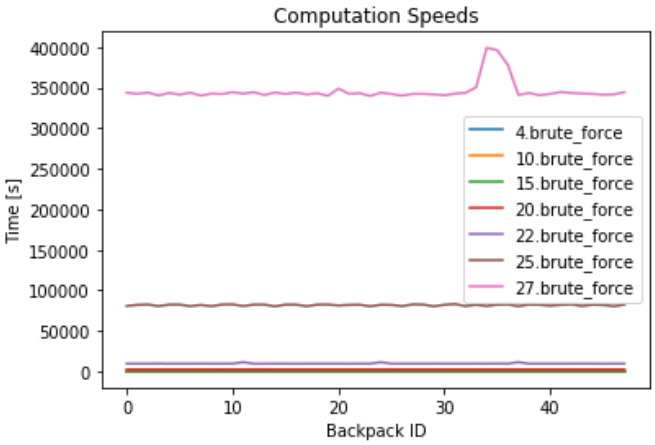
3 Měření výpočetního času

Výpočet běhu funkce je řešen tak, že je spočten strojový čas před během funkce, a také po něm. Tyto časy jsou od sebe odečteny a je vrácen čas v ms.

```
def timing(f):
    def wrap(*args):
        time1 = time.time()
        ret = f(*args)
        time2 = time.time()
        measured_time.append(
          {'type': f.__name__,
           'time': (time2-time1)*1000.0})
        return ret
    return wrap
```

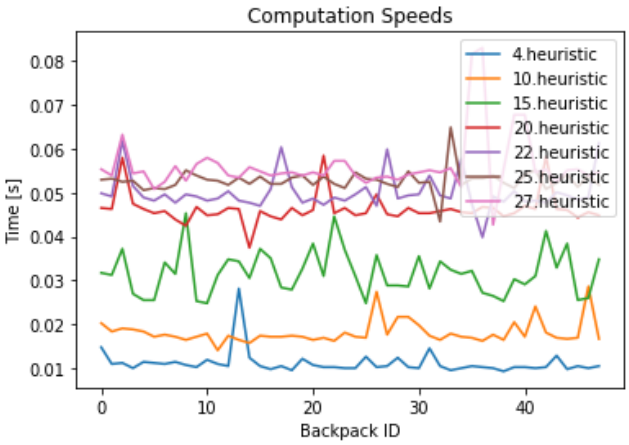
4 Brute force - závislost výpočetního času na n.

Jak můžeme pozorovat u metody hrubé síly, složitost při velikosti batohu roste exponenciálně. Data, která byla využita pro vykreslení grafů jsou k dispozici v adresáři **report/data/** v souboru **all_times.csv**.



4.1 Heuristika - závislost výpočetního času na n.

Díky tomu, že v metodě heuristiky pouze batoh sortíme a poté se snažíme přidat n objektů, můžeme říci, že složitost metody je lineární.



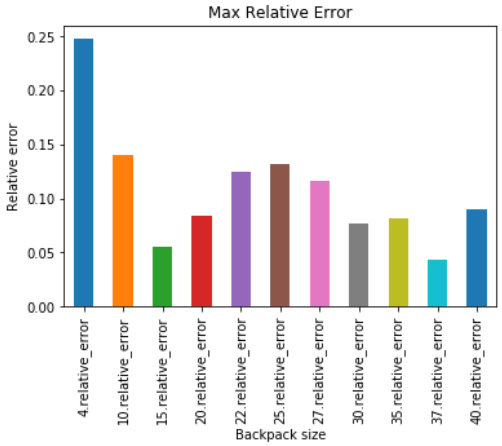
4.2 Průměrná a maximální relativní chyba

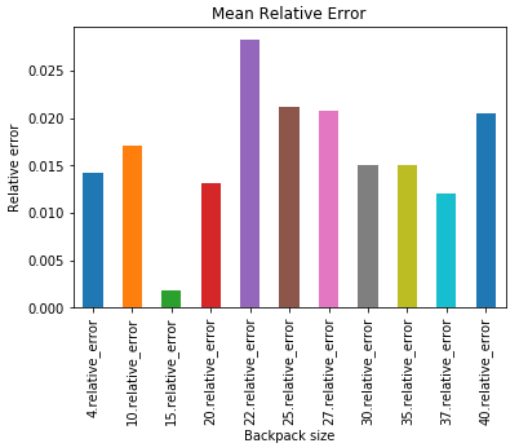
Relativní chybu jsem počítal následovně:

$$\epsilon = \frac{C(OPT) - C(APX)}{C(OPT)}$$

kde **C(OPT)** je cena optima (získaného z referenčního řešení),
a **C(APX)** je cena přibližného řešení (vypočítaná funkcí heuristic).

Relativní chyby byly vypočítány jednotlivě pro všechny instance a poté zprůměrovány (nebo vybrána maximální hodnota) pro každou velikost batohu.





Ať už z maximálních, či průměrných relaticních chyb jsem nezpozoroval žádnou závislost relativní chyby na velikosti batohu. Data, která byla využita pro vykreslení grafů jsou k dispozici v adresáři **report/data/**.

5 Shrnutí a výsledky

Pomocí metody hrubé síly jsem dosáhl pouze velikosti 27 - při velikosti bahohu již vypočtení testovacích dat trvalo více než 24 hodin. Díky tomu v grafech větší data pro měření rychlostí nejsou přiložena. Relativní chyba používá data z referenčního řešení, a díky tomu, že řešení heuristikou je mnohem rychlejší, než hrubou silou, proto data obsahuje pro všechny zadaná data.

Pro vytváření grafů bylo využito Python notebooku, který je přiložen v adresáři **report/**. Grafy jsou vykresleny pomocí knihovny **mathplotlib**.