Jan Klicpera

NI-KOP – Domácí Úkol 4

1. Specifikace úlohy

Detailní specifikace je dostupná na https://moodle-vyuka.cvut.cz/mod/assign/view.php?id=89702.

Úkolem je seznámit se s pokročilou iterativní technikou pro řešení problému batohu. Zvolil jsem simulované ochlazování.

2. Implementace heuristiky

Pro implementaci byl zvolen jazyk C#, výstupem je Windows Form aplikace, která umožňuje měnit parametry heuristiky a vstupní instance. Aplikace také vykresluje graf informující o průběhu heuristiky.

Čas běhu byl měřen pomocí systémové třídy *Stopwatch*. Při měření času je každá instance spouštěna pětkrát, výsledným časem je průměr těchto pěti běhů. Jelikož některé strategie využívají generátor náhodných čísel, tak je těchto 5 instancí vždy spouštěno se stejným seedem.

Při implementaci jsem především dbal na flexibilitu v rámci parametrizace algoritmu. Parametrizované jsou nejen hodnoty některých konstant, ale i všechny funkce, které řídí výpočet. Konkrétní implementace metod pro řízení výpočtu (cool(), equilibrium(), ...) je řešiči poskytnuta až za běhu programu, lze tedy řešiči poskytovat různé implementace a různě je kombinovat(použil jsem návrhový vzor Strategy).

Main loop

Hlavní loop heuristiky jsem implementoval velmi podobně, jako bylo předvedeno na přednášce:

```
var state = getStartState(...);
var best = state;
var temperature = startingTemperature;
while (! frozen (...)) {
        while (equilibrium(...)) {
                state = try(state);
                if (state.score > best.score)
                        best = state;
        }
        temperature = cool(...);
}
```

Konkrétní implementace funkcí, které tento loop využívá jsou popsány níže.

Funkce getStartState

Funkce, která vrací počáteční stav běhu heuristiky. Implementoval jsem dvě varianty:

- náhodný stav
- stav získaný pomocí greedy redux heuristiky

Funkce frozen

Tato funkce rozhoduje, kdy se má heuristika zastavit. Implementoval jsem dvě varianty:

- zamrznutí nastane, když je se teplota dostane pod nastavenou teplotní mez
- zamrznutí nastane, když je splněna předchozí podmínka, nebo během equilibria bylo odmítnuto $k\%$ stavů.

Funkce equilibrium

Tato funkce rozhoduje, kolik pokusů o nalezení nového stavu bude provedeno pro danou teplotu. Implementoval jsem dvě varianty podle přednášky:

- equilibrium je „ukončeno“, když počet iterací inner loopu dosáhne hodnoty N. (N je škálováno podle velikosti instance)
- equilibrium je „ukončeno“, když bylo během equilibria přijato N nových stavů. Je také ukončeno, když počet iterací inner loopu dosáhne hodnoty 2N. (zde N také škálováno)

Funkce cost

Stavy je zapotřebí porovnávat, abychom se mohli rozhodnout, zda přijmeme nový stav. Každý stav tedy musíme být schopni ohodnotit, tuto hodnotu nazveme cost. Triviálně by cost mohl být přímo hodnota optimalizačního kritéria. Pokud však chceme cestovat i do stavů, které nesplňují omezující podmínku, tak je vhodné stavy, které nejsou řešením nějakým způsobem penalizovat. Implementoval jsem pouze jeden způsob penalizace, pokud konfigurace překračuje váhu batohu, pak se od ceny odečte tato hodnota:

$$(WEIGHT - KNAP\_SIZE) * M$$

kde M je nastavitelný parametr.

Funkce try

Tato funkce určuje, jakým způsobem bude procházen stavový prostor – na momentální stav aplikuje nějaký námi navržený operátor. Takto získaný stav porovná pomocí funkce cost() s předchozím stavem. Pokud má nový stav lepší cost, pak je přijat. Pokud ne, pak je přijat s pravděpodobností

$$e^{\frac{\delta}{T}}$$

$$\delta = new.cost() - prev.cost()$$

Implementoval jsem pouze jednu variantu funkce try – přidání/odebrání náhodného předmětu z batohu (bit-flip).

Funkce cool

Tato funkce určuje, o kolik bude teplota snížena po ukončení každého equilibria. Implementoval jsem pouze jednu variantu – teplota je násobena zadaným koeficientem $q;\ 0 < q < 1$.

Ostatní vstupní parametry

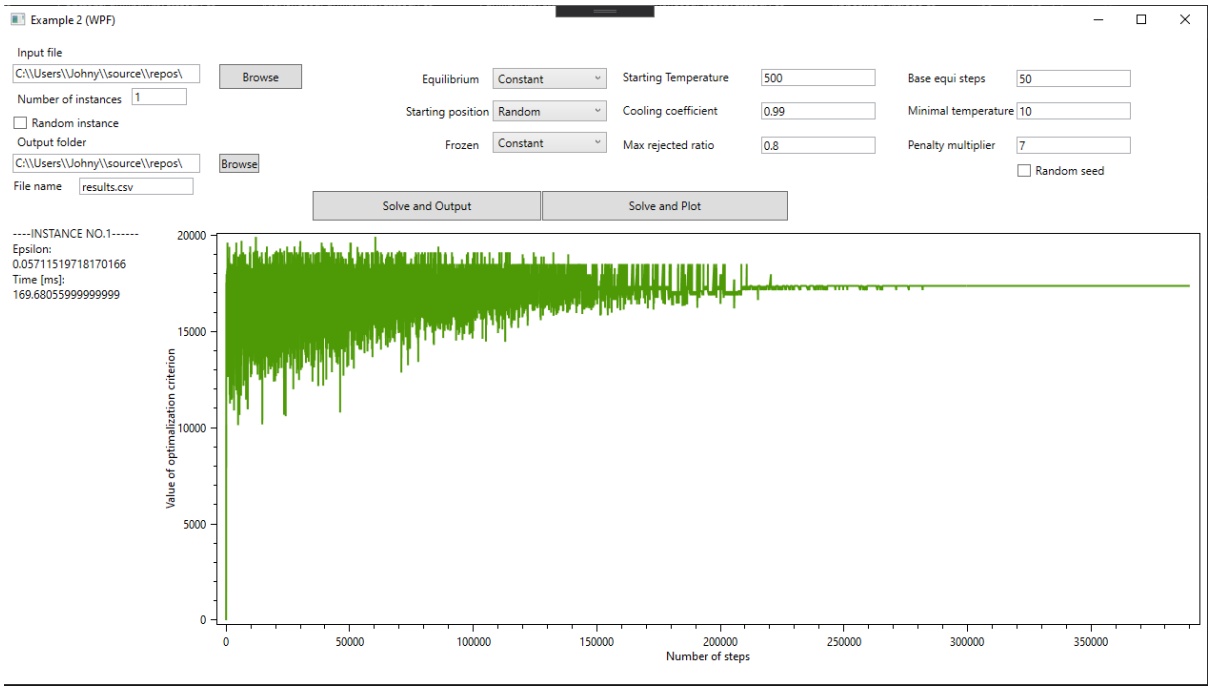
Heuristiku je dále možné ladit pomocí následujících numerických parametrů:

- Počáteční teplota
- Ochlazovací koeficient
- Maximální poměr odmítnutých stavů ku přijatým během equilibria. Tento parametr je využit pouze pro druhý výše popsaný způsob implementace metody frozen.
- Základní počet kroků každého equilibria. Tuto hodnotu heuristika pronásobí velikostí instance (výše je takto vzniklá hodnota popisována jako N)
- Minimální teplota – pokud teplota klesne pod tuto hodnotu, pak je heuristika ukončena.
- Míra penalizace překročení váhy batohu. Penalizace je vypočítána vynásobením této hodnoty váhou, o kterou byla kapacita překročena.

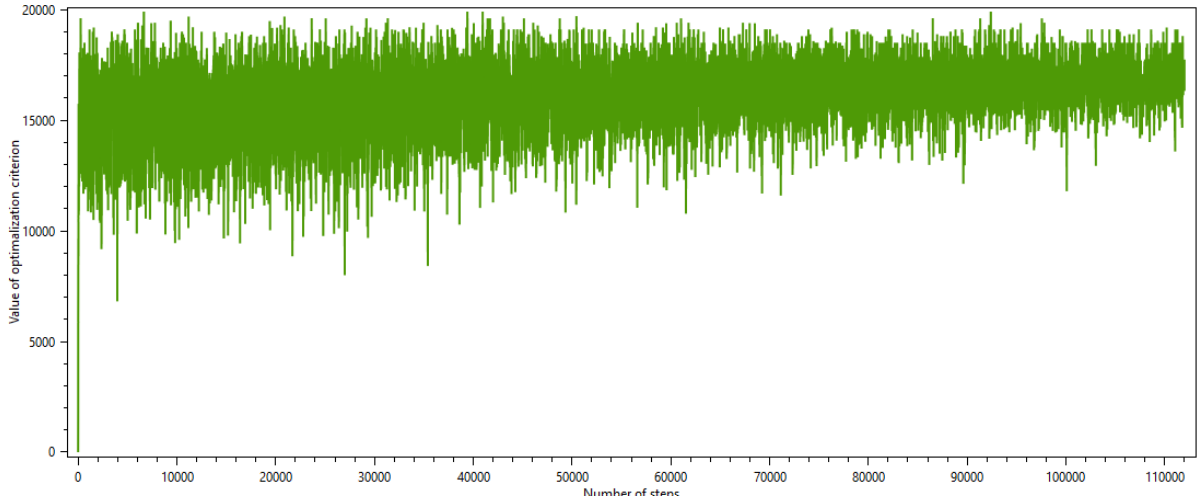
3. Ladění heuristiky a naměřené hodnoty

První iterace white box ladění

Heuristiku jsem nejprve ladil na malé sadě instancí o malé velikosti (n = 20). Ve vytvořené aplikaci jsem pozoroval závislost zadaných hodnot na běhu heuristiku (vývoj hodnoty optimalizačního kritéria momentální stavu vzhledem k iterační čítači) a výsledcích (relativní chybovost a dobu běhu). Pro představu přikládám screenshot aplikace s defaultními hodnotami:



Ze screenshotu s defaultními hodnotami je patrné, že heuristika sice konverguje, nicméně zamrznutí nastane příliš pomalu, jelikož heuristika v druhé půlce běhu již téměř neprohledává stavový prostor. Bylo by možné upravovat parametr pro minimální teplotu, nicméně flexibilnější mi přijde nastavit strategii pro zamrznutí na „ratio-based“, kdy zamrznutí nastane při odmítnutí zadaného poměru stavů během equilibria. Poměr jsem nastavil na 0.8, tedy během equilibra musí být odmítnuto 80% stavů, aby došlo k zamrznutí.



Doba běhu se díky této změně snížila více než třikrát, zatímco relativní kvalita zůstala na podobné hodnotě.

Dále jsem experimentoval s nastavením počáteční teploty. Čím větší počáteční teplota se nastaví, tím „pestřejší" je fáze diverzifikace, protože heuristika přijímá téměř jakýkoliv stav. Zároveň je však potřeba i správně nastavit koeficient chlazení, aby fáze diverzifikace netrvala příliš dlouho.

Tyto parametry je zapotřebí nastavit podle nároků kladených na rychlost výpočtu a kvalitu výsledku. Pro účely této úlohy jsem si stanovil poměrně skromné cíle – průměrná relativní chyba by měla být maximálně 0.01 a doba běhu by měla být vzhledem k takto nastavené podmínce co nejmenší.

První iterace black box testování

První black box test jsem provedl po odladění s následujícími parametry:

Equilibrium	Constant	Starting Temperature	2000	Base equi steps	100
Starting position	Greedy	Cooling coefficient	0.9	Minimal temperature	10
Frozen	Move Based	Max rejected ratio	0.85	Penalty multiplier	5

Jako testovací sadu dat jsem použil sadu NK_30 z úlohy 2. V této sadě je dohromady 500 instancí o velikosti n = 30. Heuristika dosáhla těchto výsledků:

	MIN	AVG	MAX
Doba běhu [ms]	21.21	34.32	50.73
Relativní chyba	0	0.14	0.42

Tyto výsledky bohužel nejsou uspokojivé, průměrná doba běhu je sice průměrné třikrát menší, než průměrná doba běhu algoritmu branch and bound (naměřeno v druhé úloze nad stejnými instancemi), nicméně relativní chyba dosahuje poměrně velkých hodnot. V další fázi white box ladění jsem se tedy zaměřil na instance, které heuristika řešila s velkou chybou.

Druhá iterace white box ladění a black box testování

Při pozorování jsem zjistil, že pro heuristiku jsou problematické instance mající mnoho předmětů, které se nevejdou do batohu a jen několik předmětů, které se do batohu vejdou. Přípustných stavů je poté velice málo a heuristika má problém je nalézt, protože většinu doby prozkoumává nepřípustné stavy. Vzhledem k implementovanému operátoru (náhodné přidání/odebrání) jsem tedy

zvedl penalizaci za překročení omezující podmínky na velmi vysokou hodnotu, čímž jsem prakticky navštěvování těchto stavů zakázal. Toto by neměl být problém, protože stavový prostor přípustných konfigurací problému batohu je pro tento operátor souvislý.

Dále jsem nastavil počáteční stav na výsledek greedy heuristiky. Toto sice negarantuje žádné zlepšení relativní kvality výsledku, poskytuje to však heuristice ve většinu případech dobrý startovací bod. Také jsem snížil počáteční teplotu o 500.

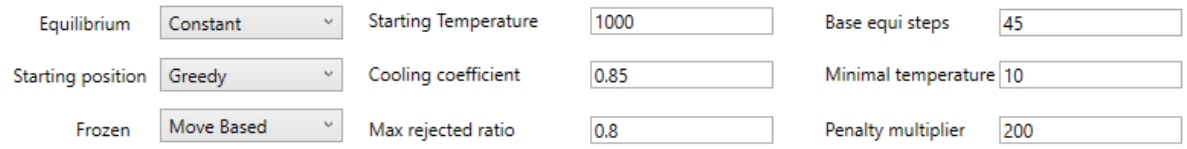
Heuristika dosáhla při těchto změnách následujících výsledků (pro sadu NK_30):

	MIN	AVG	MAX
Doba běhu [ms]	1.32	12.26	30.65
Relativní chyba	0	0.004	0.11

Heuristika již dosahuje poměrně dobrých výsledků, mnou stanovený cíl pro průměrnou relativní chybu byl dosažen, navíc klesla i průměrná doba běhu.

Třetí iterace white box ladění a black box testování

V této iteraci jsem se zaměřil na minimalizaci doby běhu při zachování maximální průměrné relativní pod hodnotou 0.01. Parametry jsem nastavil následovně:



Při použití těchto parametrů pro opětovné měření sady NK_30 bylo dosaženo těchto výsledků:

	MIN	AVG	MAX
Doba běhu [ms]	0.61	2.55	7.04
Relativní chyba	0	0.007	0.075

Heuristiku by zajisté bylo možné odladit pro dosažení ještě lepších výsledků, nicméně nepochybuji, že toho si užiji více než dost v páté úloze :).

Pomocí takto nakonfigurované heuristiky jsem řešil ještě sadu NK_35, výsledky byly následující:

	MIN	AVG	MAX
Doba běhu [ms]	0.7	2.9	9.3
Relativní chyba	0	0.0087	0.09

Je patrné, že heuristika pravděpodobně neprohledává stavový prostor úměrně ku rostoucí velikosti instance, jelikož doba běhu vzrostla pouze mírně. Nicméně i pro tuto sadu heuristika dosahuje poměrně dobrých hodnot co se týče relativní chyby. Pro tuto sadu instancí byla heuristika zhruba 350x rychlejší, než metoda větví a hranic.

4. Závěr

V rámci úlohy byl implementován řešič úlohy batohu pomocí metody simulovaného ochlazování, který poskytuje uživatelské rozhraní poskytující rychlou zpětnou vazbu o průběhu heuristiky vzhledem k nastaveným parametrům.

Efekt jednotlivých parametrů na běh heuristiky jsem v této úloze detailně neměřil, na což se plánuji zaměřit v páté úloze.

Heuristiku jsem během tří iterací odladil především experimentálně. Během tohoto měření jsem zjistil, že:

- Dobrou indikací zamrznutí je poměr nepřijatých konfigurací během daného equilibria.
- Greedy heuristika ve většině případech poskytuje poměrně dobrý počáteční stav pro heuristiku.
- Jelikož je stavový prostor přípustných konfigurací u problému batohu souvislý, tak se ukázalo být vhodné nepřípustné stavy vůbec nepřijímat.

Jako cíl pro heuristiku jsem stanovil maximální hodnotu průměrné relativní chyby na 0.01. Tento cíl se podařilo splnit, se značným snížením výpočetního času oproti exaktním algoritmům. Měření bylo však prováděno na relativně malých sadách (každá sada 500 instancí), na „obtížnější“ sadě instancí by tedy heuristika mohla dosahovat horších výsledků.