NI-KOP (cvičení)

Ondřej Wrzecionko

ZS 2022/2023

Obsah

1 Úvod **2**
1.1 Náplň cvičení 2
1.2 SAT problém 2
1.3 Problém batohu 2

2 Konfigurační proměnné **3**
2.1 Definice 3
2.2 Příklady 3
2.2.1 Minimum vertex cover 3
2.2.2 Traveling Salesman Problem 3
2.2.3 Minimum Bin Problem 3
2.2.4 Minimum Rectangle Tiling 3
2.2.5 Minimum Graph Motion Planning 3
2.2.6 The Buckets problem 4

3 Experimentální vyhodnocení algoritmu (1 instance) **4**

4 Experimentální vyhodnocení algoritmu (více instancí) **5**
4.1 ECDF 5
4.2 Porovnávání 5
4.3 Jak psát zprávu 5

5 P a NP problémy **6**
5.1 Problém Independent Set 6
5.2 Problém 3–SAT 7

6 Stavový prostor **7**

7 Nasazení simulovaného ochlazování **8**
7.1 Faktorový návrh 8

8 Nasazení simulované evoluce **9**

1 Úvod

1.1 Náplň cvičení

Odpadla konzultační cvičení, místo nich bude více cvičeních zaměřených na domácí úkoly.

Máme speciální výpočetní server ni-kop.fit.cvut.cz, na kterém je většina běžných kompilátorů, který můžeme použít pro výpočet.

Na kompendiu optimalizačních problémů lze jednotlivé problémy najít, je to i dobrá zásobárna problémů ke zkoušce.

1.2 SAT problém

Dáno: množina X n proměnných $(x_1 \dots x_n), x_i \in 0, 1$, dále Booleova formule těchto proměnných v konjunktivní normální formě o m klauzulích (součtových termech).

Rozhodnout: existuje ohodnocení Y proměnných X takové, že $F(Y) = 1$?

1	2	-3	0
1	-2	3	0
-1	-2	-3	0
-1	2	3	0

c komentář, nebo p "typ problému" "počet proměnných" "počet klauzulí"

+ znamená x_1, – znamená $\neg x_1$, 0 je vždy na konci řádku.

Zde tedy `p cnf 3 4`, jelikož jde o CNF pro 3 proměnné a 4 formule.

x	(0,0,0)	(0,0,1)	(0,1,0)	(0,1,1)	(1,0,0)	(1,0,1)	(1,1,0)	(1,1,1)
K1	1	0	2	1	2	1	3	2
K2	2	2	0	1	2	3	1	2
K3	3	2	2	1	2	1	1	0
K4	1	2	2	3	0	1	1	2

Pro každou konfiguraci je zde počet splněných literálů, 8 sloupců = 8 konfigurací, obecně těchto konfigurací je 2^n.

1.3 Problém batohu

Dáno: celé číslo n *(počet věcí)*, celé číslo M *(kapacita batohu)*, konečná množina $U = u_1, u_2, \dots, u_n$ nějakých prvků a pro každé $u \in U$: cena $c(u) \in \mathbb{N}$ a váha $w(u) \in \mathbb{N}$

Konstruktivní verze: nacpat do batohu co nejvíce věcí tak, tak aby se tam vlezly a měly co největší cenu menší než $K = 19$.

2 Konfigurační proměnné

2.1 Definice

Mám dáno ohodnocení konfiguračních proměnných = **konfigurace**. Nezaměňovat se vstupními a výstupními proměnnými.

Omezení počítám ze vstupních proměnných a dosazuji do něj konfiguraci. **Optimalizační kritérium** se počítá z výstupních proměnných, ty vychází z konfigurace. Každé řešení musí mít konfiguraci, ze které vychází, jinak se šidím.

2.2 Příklady

2.2.1 Minimum vertex cover

Hledáme minimální množinu vrcholů $V' \subseteq V$ takovou, aby z každé hrany $(u, v) \in E$ alespoň jeden vrchol ležel ve V'. Optimalizační kritérium je $|V'|$. *(kardinalita množiny vrcholů)*

$\rightarrow$ Konfigurační proměnné jsou podmnožina V. Tento problém také patří do kategorie minimální podmnožina.

2.2.2 Traveling Salesman Problem

Máme zadáno množinu C z m měst a vzdálenosti mezi nimi $d(c_i, c_j) \in \mathbb{N}$. Hledáme permutaci měst, aby vzdálenost měst byla minimální.

$\rightarrow$ Konfigurační proměnnou jsou navštívená města (pořadí). Omezující podmínkou je to, aby se jednalo o permutaci.

2.2.3 Minimum Bin Problem

Máme zadánu množinu U předmětů, který má každý velikost $s(u) \in \mathbb{Z}^+$ a kapacita krabičky B. Hledáme rozdělení U do disjunktních množin $U_1, U_2, \ldots, U_m$ takové, aby se každá množina vlezla do krabičky. Optimalizačním kritériem je množství použitých krabiček.

$\rightarrow$ Konfigurační proměnnou je dvojrozměrné pole krabiček a předmětů v nich /nebo/ pole předmětů s informací o tom, ve které krabičce jsou.

2.2.4 Minimum Rectangle Tiling

Máme pole $n \times n$ nezáporných čísel a kladné celé číslo p. Řešením je rozdělení tohoto pole na p nepřesahujících podpolí ve tvaru obdélníku. Optimalizačním kritériem je váha čísel v obdélníku.

$\rightarrow$ Konfigurační proměnnou jsou jednotlivé obdélníky, reprezentovat je můžeme jako souřadnice levého horního a pravého dolního rohu každého obdélníku.

2.2.5 Minimum Graph Motion Planning

Máme dán graf G, počáteční pozici robota, cílovou pozici robota, a pozice jednotlivých překážek. V každém tahu můžeme pohnout buď s robotem, nebo s překážkou a posunout ho/ji na sousední vrchol. Řešením je posloupnost pohybů robota a překážek. Optimalizačním kritériem je počet těchto pohybů.

$\rightarrow$ Konfigurační proměnnou je posloupnost provedených kroků (co, kam).

2.2.6 The Buckets problem

Máme n kbelíků, kohoutek a umyvadlo. Známe kapacity kbelíků, jejich původní a cílové naplnění vodou. V každém kroku lze zaplnit kyblík po rysku, vyprázdnit kyblík nebo přelít vodu z jednoho do druhého. Řešením je posloupnost přechodů, optimalizačním kritériem je počet operací.

$\rightarrow$ Konfigurační proměnnou je posloupnost přechodů. (odkud, kam) kde odkud a kam může být číslo kbelíku, T (kohoutek) nebo S (umyvadlo).

(Tento problém si můžu reprezentovat jako n rozměrný graf a konfigurační proměnnou je jakási čára po bodech.)

3 Experimentální vyhodnocení algoritmu (1 instance)

Algoritmus GSAT: snaží se o řešení problému splnitelnosti Booleovských formulí; vygeneruje náhodné rozložení proměnných, a pak se snaží ho vylepšit.

Náhodný krok – náhodný výběr nesplněné klauzule a proměnné v ní střídající se s **greedy** krokem – vybere se náhodně jedna z nejvhodnějších proměnných pro flip *(která splní nejvíce formulí)*.

Klauzule	(1,0,1)	(0,0,0)	(0,1,0)	(1,1,0)
$x_1 + x_2 + x_3$	2	0	1	2
$\neg x_1 + \neg x_2 + x_3$	2	2	1	0
$x_1 + \neg x_2 + x_3$	3	1	0	1
$\neg x_1 + \neg x_2 + \neg x_3$	1	3	2	1
$x_1 + \neg x_2 + \neg x_3$	2	2	1	2
$\neg x_1 + x_2 + x_3$	1	1	2	1
	splněno	nespl.	flip x_2	greedy

V prvním případě bylo rovnou **splněno**, v druhém zkusíme **náhodný** krok = flipneme náhodně proměnnou, abychom splnili klauzuli (zde x_2). V třetím zkusíme **hladový** krok = vybereme nejlepší možnost *(všechny možnosti zde ale splní pořád jen 5, tedy náhodně)*.

Program GSAT můžeme spustit s různými parametry: -r time iniciuje pseudonáhodný generátor, -i 1000 nastaví počet MAX_FLIPS na 1000, -p 0.4 nastaví pravděpodobnost na 0.4.

Můžeme porovnávat také pomocí **ECDF** (distribuční funkce) – viz. přednáška, zaneseme graf, je to porovnání pravděpodobností, že algoritmus skončil nejvýše v daném kroku *(včetně neúspěšných běhů)*.

4 Experimentální vyhodnocení algoritmu (více instancí)

4.1 ECDF

Mám počet kroků algoritmu (na ose x) a pravděpodobnost, že algoritmus skončil v nejvýše daném kroku. Připomenutí vzorce:

$$\sum_{k:x_k \leq x} P(X = x_k) \tag{1}$$

Křivku počítáme jen z úspěšných běhů, ale dělíme ji počtem všech běhů. Vzniká tak tedy pravděpodobnost, že algoritmus **úspěšně** skončí do kroku k.

Average fined par10 je tzv. penalizovaný průměr:

$$\frac{\sum k_i + 10 \cdot l}{n} \tag{2}$$

kde k_i je počet kroků v úspěšném běhu, l je limit kroků a n je počet běhů.

4.2 Porovnávání

Můžeme porovnávat podle parametrů σ^2, μ normálního rozložení; počtu kroků; počtu penalizovaných kroků; nebo také podle xing, winner – najde se poslední průsečík grafů ECDF a od toho se počítá kdo je nahoře.

4.3 Jak psát zprávu

Řekneme, který algoritmus je lepší na těžkých instancích o 20 proměnných (na základě porovnání z měření, co jsme dělali na cvičeních).

Material: standardní implementace gSATu; datové sady ze SATLIBu, uf20-91 (1 000 instancí); primární data (počet iterací, počet splněných klauzulí); 500 iterací max; 1 000 spuštění na 1 instanci; pilotní experiment se 100 spuštěními

Results: co jsme naměřili, k čemu jsme došli (příloha) – pilotní experiment → odvozené metriky (četnost úspěchu); celý experiment: metrika – par10, poslední křížení

Discussion: interpretace dat, zhodnocení významu odpovědi, účinnosti zvolené metody. *(avg fined steps říká, že je větší pravděpodobnost, že B dojde k správnému výsledku, než A, kratší očekávaná doba do úspěšného řešení ;; křížení: při počtu kroků větším než XY je u B větší pravděpodobnost úspěchu)*

5 P a NP problémy

5.1 Problém Independent Set

Máme graf $G(V, E)$ a celé kladné číslo m. Cílem je rozhodnout, zda-li existuje $V' \subset V$ taková, že mezi každými $u, v \in V'$ není hrana a $|V'| \geq m$.

Jsme schopni v polynomiálním čase **ověřit** certifikát, problém tedy leží v **NP**. **Komplementární** problém: platí pro každou podmnožinu s méně než m prvky, že mezi nimi je alespoň jedna hrana? Svědkem by zde byly **všechny** tyto podmnožiny, ty se ale dají reprezentovat drobně.

Je tento problém **NP-úplný**? Pokud ano, pak by byl NP-těžký a v NP. Dokázat, že je NP-těžký můžeme tak, že dokážeme polynomiálně převést **SAT na** tento problém. *(pak to vyplývá z Carpovy redukce díky tomu, že je to tranzitivní; cokoliv $\to$ SAT $\to$ můj problém)*

Co to znamená **převést problém**? Převést **každou** instanci SAT (formule) na instanci IS (graf + číslo) v **polynomiálním** čase tak, že formule je splnitelná, **právě když** $\exists$ nezávislá množina velikosti $\geq M$.

The Reduction

1. G_φ will have one vertex for each literal in a clause
2. Connect the 3 literals in a clause to form a triangle; the independent set will pick at most one vertex from each clause, which will correspond to the literal to be set to true
3. Connect 2 vertices if they label complementary literals; this ensures that the literals corresponding to the independent set do not have a conflict
4. Take k to be the number of clauses

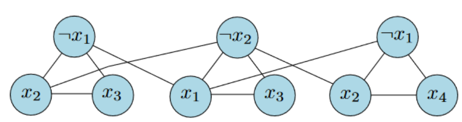


Figure: Graph for $\varphi = (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_4)$

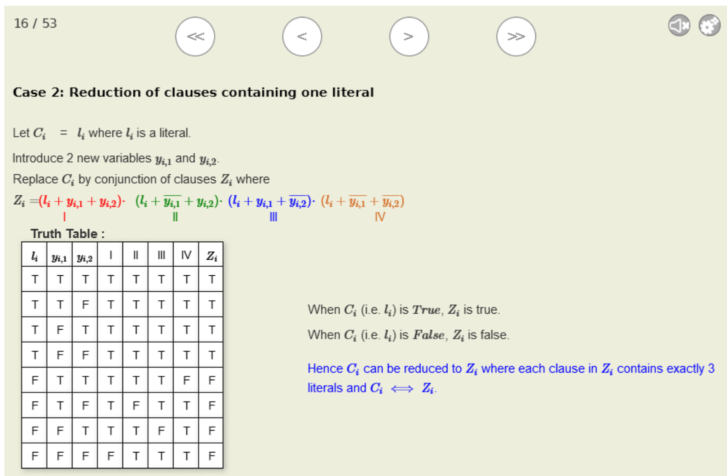
25 / 63

Vidíme, že zde půjde každý krok převést v polynomiálním čase. Platí ale naše ekvivalence? Pokud je formule splnitelná, pak platí v každé klauzuli min. 1 literál a díky konstrukci vím, že za každou klauzuli bude v cílové množině **právě jeden** uzel, budou tedy 3. Tak stejně obrácený směr: z konstrukce vidíme, že to bude opravdu splnitelné. Podrobný důkaz zde.

5.2 Problém 3–SAT

Jak budeme konstruovat Karpovu **redukci**? Vstupem bude SAT formule a výstupem bude 3–SAT formule. Potřebuji vyřešit správně počet literálů na vstupu.

Když mám tři literály, je to ok. Když jsou dva $(b + c)$, převedu to na $(b + c + x), (b + c + \neg x)$. Když je jeden, přidám tam x_1, x_2 ve 4 variantách. Když jich je více: např. 4 – rozdělím na $(a + b + x_1), (c + d + x_2)$. Náčrt důkazu zde.



6 Stavový prostor

Co jsou **konfigurační proměnné** u SATu? Co je jeho certifikát? Jsou to **ohodnocení** Y jednotlivých proměnných $x_1, \ldots, x_n$, tedy funkce $y_1 = Y(x_1), \ldots$

Co je **stavový prostor** algoritmu A řešícího instanci I problému Π? Je to **uspořádaná dvojice** stavů a operátorů, které umožňují přechod mezi nimi. U SATu jsou stavy **ohodnocení** jednotlivých **konfiguračních proměnných** (x_m) a operátory budou **funkce**, které dané proměnné změní stav (např. $Y(x_m) = \neg x_m$).

Tento graf u GSATu je **silně souvislý**, bude vypadat jako n-rozměrná krychle, mezi jednotlivými stavy se přesunu v m krocích, kde m je Hammingova vzd.

Už známe hledání typu BFS, DBS, nebo podle prioritní fronty *(prioritou je hodnota optimalizačního kritéria)*. To je **systematické** prohledávání, nebo dokonce **úplné**.

Existují ale způsoby, které prostě můžou skončit v nějakém "lokálním minimu" – best only, first improvement…

Hra **sokoban**: stavem nejsou polohy beden, ale sekvence pohybů beden.

7 Nasazení simulovaného ochlazování

Když děláme simulované ochlazování, začneme tím stejným, co u lokálních heuristik: stavový prostor.

U SATu jde tedy o stav – ohodnocení jednotlivých proměnných, operátorů bude n a každý z nich neguje svou proměnnou. Tento prostor bude **symetrický** – každý operátor je sám sobě inverzní a každý stav je **dosažitelný**.

U **rozvrhu ochlazování** bude záležet na náhodné volbě souseda – vyberu náhodně operátor (proměnnou) a zkusím ji flipnout. Pokud je po flipnutí stav lepší (více splněných klauzulí), beru toto řešení vždy, pokud je stav horší, s určitou pravděpodobností závislou na **zhoršení** a **teplotě** se toto řešení může vzít.

Počet splněných klauzulí ... $E(s)$, platí, že $\delta = E(s_1) - E(s_2)$, pravděpodobnost je pak $p = e^{-\frac{\delta}{T}}$. Jako metodu **ochlazování** volíme ono $T(x) = T_p \cdot \alpha^{\frac{x}{N}}$, kde N je délka ekvilibra.

7.1 Faktorový návrh

Faktorový návrh: vyberu všechny parametry, kterým měním hodnoty, udělám jejich kartézský součin, a zkoumám.

Ekvilibrium zde volíme pevné, koeficient bude souviset s ním. Volím tedy např. $\alpha = 0.95, T = 20$ jako střední hodnoty a zkusím zkoumat rozsah $\alpha = 0.8$ až $\alpha = 0.99$ a $T = 5$ až $T = 50$.

Na začátku si vybíráme koeficient chlazení α, (délku ekvilibria N) a počáteční teplotu T, uděláme ten kartézský součin a zapíšeme počty běhů do tabulky. Pak pozoruji.

Z pozorování vidíme, že je lepší volit **nižší** počáteční teplotu $T = 5$ nebo 3, nemá ale zas tak důležitou roli *(aktuálně, u SAsatu)*, koeficient ochlazování určuje **nepřímou** úměru počet kroků vs počet úspěšných běhů. Naše závěry této white-box fáze bychom měli pak ověřit na sadách (black-box).

Z pozorování u SAsatu tedy vidíme, že pro $n = 20$ stačí $\alpha = 0.95$, aby bylo vše vyřešeno, u $n = 50$ to je už $\alpha = 0.999$, $n = 100 \rightarrow \alpha = 0.99999$ ...

Nejprve jsme udělali faktorový návrh na **jedné** instanci s možnými parametry, z toho jsme zjistili nějakou **počáteční** teplotu, tu jsme ověřili na jiných instancích a určili jsme **strategii** řízení teploty (konstanta, nebo určíme estimátorem). U řízení α jsme zjistili na čem **závisí** na lehkých sadách a určili strategii řízení ochlazování = konec a závěr **white-box** fáze.

Black-box fáze je již to, co jsme dělali v 1. domácím úkolu.

8 Nasazení simulované evoluce

Způsob výběru v genetických algoritmech = **selekční tlak** (pravděpodobnost výběru **nejlepšího** jedince). Zdatnost jedince je určena hodnotou **optimalizačního** kritéria. Pokud nám to tedy nechtělo konvergovat, měli jsme možná malý selekční tlak.

Pokud mám na začátku **malé rozdíly** v populaci, může to být důvodem malého selekčního tlaku a pomalé konvergence. Řešením je použití **lineárního škálování** → méně časté hodnoty se "naškálují" *(funguje oběma směry – při malých i velkých rozdílech)* tak, že **nejméně** zdatný jedinec bude mít **zdatnost** z_1 a **nejvíce** z_2 (slide KOP09-33). Pravděpodobnost **mutace** se typicky volí jako **nízké** jednotky procent (např. 3 %).

Kromě **lineárního škálování** můžu u výběru ruletou určovat selekční tlak ještě **ranking**em = nejhorší jedinec má zdatnost 1 a každý lepší o jedna větší, tím získáme pevný selekční tlak. (slide KOP09-38) nebo **zkráceným výběrem** (pracuji jen s lepší polovinou).

V případě turnaje je **velikost turnaje** přímo úměrná selekčnímu tlaku. Při **faktorovém návrhu** bychom měli velikost populace, pravděpodobnost mutace, horní mez lineárního škálování *(někdy je třeba trochu jemnější škála)*.