Systémy pro řešení omezujících podmínek

Michal Tuláček

Seminář z umělé inteligence I

Agenda

Sudoku

2	5			3		9		1
	1				4			
4		7				2		8
		5	2					
				9	8	1		
	4				3			
			3	6			7	2
	7							3
9		3				6		4

Problém s omezujícími podmínkami

- Množina proměnných x_i
- Domény proměnných D_i ($x_i \in D_i$)
- Podmínky na proměnných

Označuje se zkratkou *CSP* (constraint satisfaction problem)

Řešení CSP

- Řešení CSP je takové ohodnocení proměnných x_i, že jsou splněny všechny podmínky
- Ne každý problém musí mít řešení
- Pokud zadáme objektivní funkci, můžeme řešit optimalizační problém

Metody řešení CSP

- Inference
  - Konzistence, NC, AC, PC
- Prohledávání
  - Backtrack
  - First fail
  - Look ahead, forward checking
- Optimalizace
  - Branch and bound

Pokud chceme použít techniky řešení CSP ve svém programu, buď si tyto techniky implementujeme sami a nebo použijeme systém pro řešení omezujících podmínek.

Systém pro řešení omezujících podmínek

- Samostatný program nebo knihovna
- Vstupem je model problému
- Výstupem je řešení problému, tedy ohodnocení proměnných vyhovující podmínkám
- Zpravidla řešení na klíč a black-box

Jaký systém zvolit

- O tom je zbytek této přednášky
- Porovnáme šest systémů pro různé systémy a programovací jazyky
- Zhodnotíme náročnost proniknutí do řešiče
- Podrobíme systémy benchmarkům
- Podrobněji v mé bakalářské práci

Agenda

N královen

Rozmístěte n královen na šachovnici o velikosti $n \times n$ tak, aby se navzájem neohrožovaly.
Model:

- Proměnné a domény:
  - $q_1, ..., q_n \in \{1, ..., n\}$.
- Podmínky:
  - $\forall i, j \in \{1, ..., n\} : q_i \neq q_j$,
  - $\forall i, j \in \{1, ..., n\} : |q_i - q_j| \neq |i - j|$
  - $q_1 < q_n$

Magická sekvence

Posloupnost $x_0, x_1, ..., x_{n-1}$ je magická, pokud se číslo na pozici i vyskytuje v sekvenci právě n_i-krát.
Například $(2, 1, 2, 0, 0)$.
Model:

- Proměnné a domény:
  - $m_0, ..., m_{k-1} \in \{0, ..., k\}$.
- Podmínky:
  - $\forall i \in \{0, ..., k-1\} : m_i = \sum_{m_j=i} 1$.

Sebereferenční kvíz

1. The first question to which the answer is A:
(A) 4 (B) 3 (C) 2 (D) 1 (E) none of above
2. The only two consecutive questions with identical answers:
(A) 3 and 4 (B) 4 and 5 (C) 5 and 6 (D) 6 and 7 (E) 7 and 8
3. The next question with answer A:
(A) 4 (B) 5 (C) 6 (D) 7 (E) 8
4. The first even numbered question with the answer B:
(A) 2 (B) 4 (C) 6 (D) 8 (E) 10
5. The only odd numbered question with the answer C:
(A) 1 (B) 3 (C) 5 (D) 7 (E) 9
6. A question with answer D:
(A) comes before this one, but not after this one (B) comes after this one, but not before this one (C) comes before and after this one (D) does not occur at all (E) none of the above
7. The last question to which the answer is E:
(A) 5 (B) 6 (C) 7 (D) 8 (E) 9
8. The number of questions to which the answer is a consonant:
(A) 7 (B) 6 (C) 5 (D) 4 (E) 3
9. The number of questions to which the answer is a vowel:
(A) 0 (B) 1 (C) 2 (D) 3 (E) 4
10. The answer to this question is:
(A) A (B) B (C) C (D) D (E) E

Sebereferenční kvíz – řešení

Question	A	B	C	D	E
1	0	0	1	0	0
2	1	0	0	0	0
3	0	1	0	0	0
4	0	1	0	0	0
5	1	0	0	0	0
6	0	1	0	0	0
7	0	0	0	0	1
8	0	1	0	0	0
9	0	0	0	0	1
10	0	0	0	1	0

QWH – Kvazigrupy s dírami

Doplňte částečně vyplněný latinský čtverec o rozměrech $n \times n$
Jedná se o variantu QCP, ale má vždy řešení. Obtížnost závisí na velikosti *páteře* čtverce.
Model:

- Proměnné a domény:
  - $q_{11}, ..., q_{nn} \in \{1, ..., n\}$
- Podmínky:
  - $\forall i \in \{1, ..., n\} : \forall j, k \in \{1, ..., n\} : q_{ij} \neq q_{ik}$
  - $\forall i \in \{1, ..., n\} : \forall j, k \in \{1, ..., n\} : q_{ji} \neq q_{ki}$
  - data_{ij} definováno $\Leftrightarrow (q_{ij} = \text{data}_{ij})$

Sklady

Optimalizační úloha. Máme dané prodejny a chceme otevřít sklady, které je budou zásobovat. Máme určit, které sklady otevřeme za předpokladu, že každý sklad má pevně danou kapacitu (počet prodejen, které je schopen obsloužit) a každá prodejna musí být zásobena právě jedním skladem. Cílem je minimalizovat celkovou cenu vynaloženou na projekt, přičemž cena za otevření skladu je fixní a cena dopravy sc_{ij} ze skladu i do skladu j je daná v matici.

Sklady – model

- Proměnné a domény:
  - $\text{totalCost} \in \mathbb{N}$
  - $\text{numberOpen} \in \{0, W\}$
  - $\text{open}_1, ..., \text{open}_W \in \{0, 1\}$
  - $\text{supplier}_1, ..., \text{supplier}_S \in \{1, ..., W\}$
  - $\text{cost}_1, ..., \text{cost}_S \in \mathbb{N}$
  - $\text{costSum} \in \mathbb{N}$
- Podmínky:
  - $\text{totalCost} = \text{costSum} + \text{numberOpen} \cdot \text{openCost}$
  - $\text{costSum} = \sum_i \text{cost}_i$
  - $\text{numberOpen} = \sum_i \text{open}_i$
  - $\forall i \in \{1, ..., W\} : w_i \geq \sum_{\text{supplier}_j = i} 1$
  - $\forall i \in \{1, ..., W\} : (\text{open}_i = 1) \Leftrightarrow \left(\left(\sum_{\text{supplier}_j = i} 1\right) > 0\right)$
  - $\forall i \in 1, ..., S, \forall j \in \{1, ..., W\} : (\text{supplier}_i = j) \Rightarrow (\text{cost}_i = \text{supplyCost}_{ij})$

Mozart/Oz

- Oz – multiparadigmatický programovací jazyk (procedurální, funkcionální, oop, logické programování, paralelizmus, csp, ...)
- Prostředky pro řešení CSP jsou přímo součástí jazyka
- Dobře dokumentovaný
- "Divoká" syntax
- Space, constraint store, propagátory
- možnost napsat si vlastní propagátor
- IDE Emacs, má kompilátor, umožňuje vytvářet applety pro webové stránky
- Debug - Oz Explorer
- Pravděpodobně mrtvý (žádná aktivita více než přes rok, včetně bugfixů)

Agenda

Choco

- Knihovna pro Java
- Odděluje modelování a hledání
- Uživatel nadefinuje model (objekt, kterému nastaví vlastnosti) a ten předá řešiči, který nalezne řešení
- Podporuje jak definici vlastních podmínek tak vlastní prohledávací strategie
- Referenční dokumentace v JavaDoc
- Dokumentace na webu – je tam vše, ale jen pokud víte kde hledat
- Mailová konference (živá, velmi nízká reakční doba, rozumné reakce)
- Debug – běhový log

Minion

- Samostatný program, vlastní formát vstupu, výstup na stdout
- "Podmínkový assembler"
- Omezená sada podmínek
- Podmínka $\sum x_i = c$ je syntaktický cukr, místo toho je nutné použít $\sum x_i \leq c \wedge \sum x_i \geq c$
- Není možní přidávat vlastní podmínky
- Bez generátoru nepoužitelné
- Tailor
- S generátorem vhodné např. i jako CSP systém pro bashové skripty
- Dokumentace dostatečná
- Debug – běhový log
- Nejrychlejší solver (podle autorů)

Gecode

- Knihovna pro C++
- Definice modelu je zakódovaná ve zdrojovém kódu třídy
- Podporuje jak definici vlastních podmínek tak vlastní prohledávací strategie
- (Konečně) velmi obsáhlá tutoriálová dokumentace
- Emailová konference (opět velmi živá)
- Podporuje jak definici vlastních podmínek nat vlastní prohledávací strategie
- Debug - ekvivalent nástroje Oz Explorer

ECL^{i}PS^e

- Open source implementace Prologu
- Řešič dodaný ve formě knihovny *ic*
- Umí řešit nejen CSP v $\mathbb{Z}$ ale i v $\mathbb{R}$ a podporuje také podmínky na celočíselné množiny
- Podpora pro tvorbu vlastních podmínek
- Debug – vizualizace přes *viewable* knhovnu, možno zaregistrovat nqad danými proměnnými příslušný aplet
- Dokumentace je vyčerpávající, jak pro CSP řešič tak pro celý Prolog
- Méně podmínek než ostatní systémy

SICStus Prolog

- Jediný komerční systém ve srovnání
- Prolog
- Podporuje více řešičů: $clp(fd)$, $clp(b)$, $clp(q, r)$
- Nepodporuje celočíselné množiny
- Syntax pro popis modelu je kompatibilní s ECL^{i}PS^e, takže jde model jednoduše přepsat s minimem změn
- Debug – knihovna *fdbg* nabízí detailní pohled do činosti solveru
- Dokumentace podobně vyčerpávající jako u ECL^{i}PS^e
- Ke stažení trial verze

Výkonnostní testy

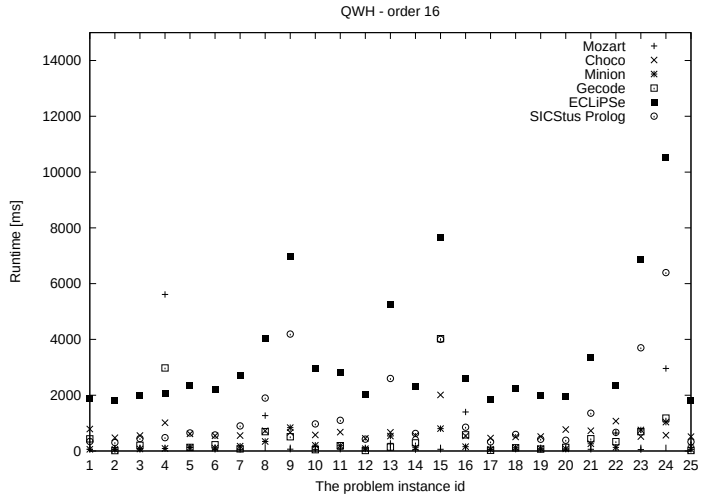
Tabulka: Runtime (ms)

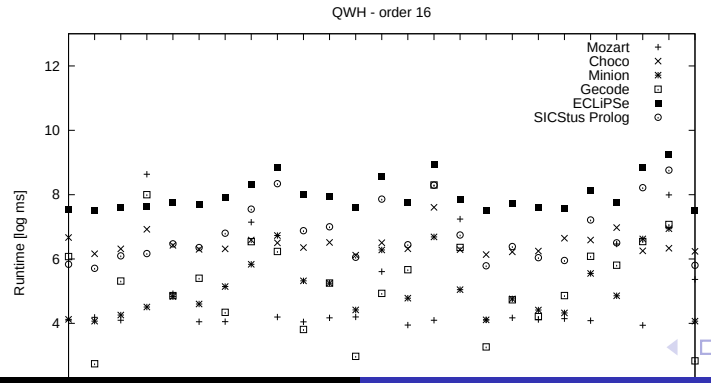
Benchmark	Moz	Cho	Min	Gec	ECL	SiP
queens10	237.95	850.27	227.69	**65.35**	809.45	327.53
queens100	184.19	1005.02	×	34.70	×	**15.40**
magic20	**11.48**	1092.95	71.24	65.41	4514.14	198.32
srq	**6.22**	482.81	55.81	25.89	691.13	13.54
warehouses	68.28	1500.83	2338.45	169.71	1808.29	**14.41**

Tabulka: Memory peak (MB)

Benchmark	Moz	Cho	Min	Gec	ECL	SiP
queens10	8.62	209.42	44.17	**8.58**	1158.15	38.08
queens100	23.31	209.46	×	6.71	×	**1.35**
magic20	**2.32**	1173.19	44.57	7.26	1157.99	39.30
srq	63.36	209.35	44.56	6.68	1158.08	**0.94**
warehouses	6.79	209.12	44.30	7.00	1157.99	**0.89**

Výkonnostní testy





Testy robustnosti

Jak dlouhou magickou sekvenci umí řešič spočítat v daném časovém limitu?

Moz	Cho	Min	Gec	ECL	SiP
94	111	**236**	191	55	174

Agenda

Závěr

- Všechny řešiče mají své pro a proti
- Můj osobní výběr je Gecode, SICStus Prolog, Choco (v tomto pořadí)
- Ale záleží na úloze (prog. jazyk, ochota použít komerční řešič, ...)

Děkuji za pozornost

Otázky?