Задачи удовлетворения ограничений. Программирование в ограничениях.

Ирина Броварь

21 марта 2012 г.

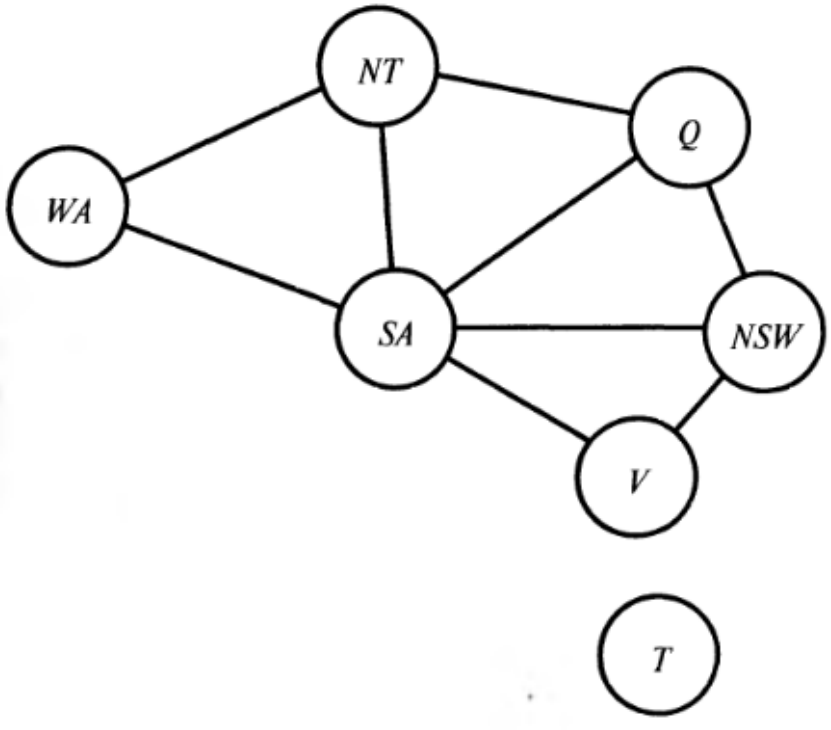
Задачи удовлетворения ограничениям

Constraint Satisfaction Problem (CSP)

- Множество переменных $X_1, X_2, ..., X_n$ с областью определения
- Множество ограничений $C_1, C_2, ..., C_m$ — подмножества переменных и допустимые комбинации значений переменных из этих подмножеств
- **Решение задачи CSP** - полное(для каждой переменной) присваивание, удовлетворяющее всем ограничениям.
- Возможно дополнительное условие - максимизация целевой функции.

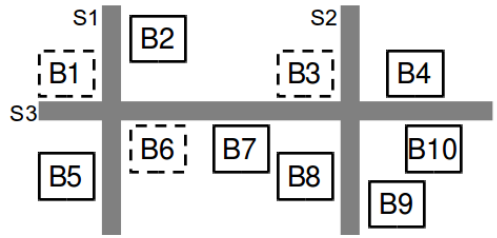
Представление задачи CSP

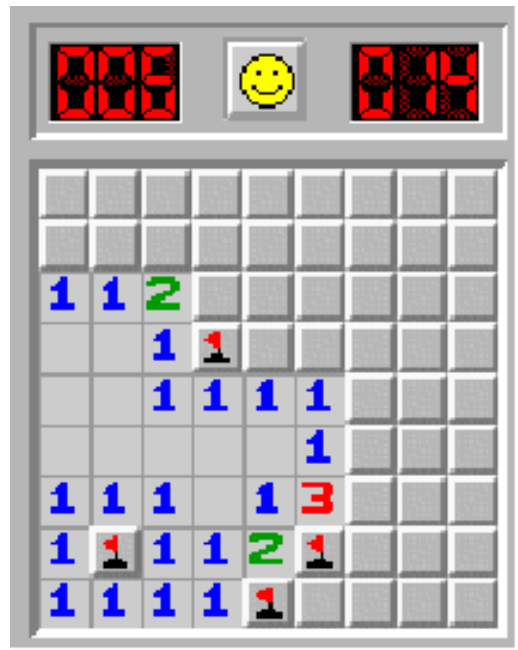
Задачу CSP можно представить в виде графа ограничений.
Узлы - переменные, дуги - ограничения.



Примеры применения

- Задача раскраски карты
- Задача о расстановке ферзей
- Головоломки(числовые ребусы, задача Эйнштейна, судоку, быки и коровы)
- Составление расписания
- Распределение ресурсов
- Определение адреса здания
- Обработка естественного языка
- Задача SAT — задача проверки выполнимости формулы логики высказываний





Поиск с возвратом

Инкрементная формулировка

1. Начальное состояние — $\{\}$
2. Функция определения преемника — $\{X_i = v_i\}$ (при условии, что новая переменная не конфликтует с присвоенными ранее)
3. Проверка пути — является ли текущее присваивание полным
4. Стоимость пути — постоянная стоимость для каждого этапа

Путь, по которому достигается решение, не представляет интереса.

Эвристики - 1

- **Упорядочивание переменных и значений**

  1. Эвристика с минимальным количеством оставшихся значений (MRV — Minimum Remaining Values)

  2. Степенная эвристика — выбор переменной, участвующей в наибольшем количестве ограничений на другие переменные с неприсвоенными значениями

  3. Эвристика с наименее ограничительным значением — исключается наименьшее количество вариантов выбора значений для соседних переменных в графе ограничений

Эвристики - 2

- **Распространение информации с помощью ограничений**
  1. Предварительная проверка — проверка на несовместимость соседних значений
  2. Распространение ограничения — распространение последствий применение одного ограничения на другие переменные
  3. Проверка совместимости дуг — дуга (x,y) является совместимой, если для каждого значения x существует значение y, совместимое с x
- **Обработка специальных ограничений**
  1. Все различные значения(alldiff)
  2. Ресурсное ограничение(atmost)
  3. Распространение пределов

Поиск в обратном направлении

Идея: Вернуться к одному из множеств переменных, которые стали причиной неудачи — конфликтных множеств.
Можно показать, что каждая ветвь, отсекаемая с помощью обратного перехода, отсекается также с помощью предварительной проверки.

$$conf(X_i) \leftarrow conf(X_i) \cup conf(X_j) - \{X_i\}$$

Модификация — **метод определения ограничений с помощью обучения** — добавление нового ограничения, выдвинутого на основе логического анализа этих конфликтов.

Применение локального поиска

В начальном состоянии присваивается значение каждой переменной, а функция определения преемника действует по принципу изменения за один раз значения одной переменной. В задачах с целевой функцией можно применять метод отжига.

Эвристика с минимальными конфликтами — для выбора нового значения переменной.

Используется для оперативной корректировки в случае изменения условий задачи.

Что такое Пролог

Пролог —дедуктивная система, которая находит ответы на **запросы**, используя базу знаний, состоящую из **фактов** и **правил**.

Декларативное программирование

- Создание базы данных фактов и правил
- Интерпретатор Пролога дедуктивно выводит ответ

Структура программы на Прологе

- Исходный код *.pl
- База данных
- Запросы

base.pl

```
1  sol(X) :-
2    =(X, [F, T, U, W, R, O]),
3    domain([F, T], 1, 9),
4    domain([U, W, R, O], 0, 9),
5             100*T + 10*W + O +
6             100*T + 10*W + O =:=
7    1000*F + 100*O + 10*U + R,
8    all_different(X).
9
10 domain([], _, _) :- true.
11 domain([X | Xs], Start, Stop) :-
12     between(Start, Stop, X),
```

```
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 5.10.4)
Copyright (c) 1990-2011 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?- sol(X).
X = [1, 7, 3, 6, 0, 5] ;
X = [1, 7, 6, 3, 8, 4] ;
X = [1, 8, 3, 6, 4, 7] ;
X = [1, 8, 7, 3, 2, 6] ;
X = [1, 8, 9, 4, 2, 6] ;
X = [1, 9, 5, 2, 6, 8] ;
X = [1, 9, 7, 3, 6, 8] ;
false.

?-
```

Термы

- Атомы(atom, y_123, 'Atom')
- Переменные(X, _name, _)
- Структуры(functor(arg1,...,argn))
- Списки([1,2])

Факты в Прологе

Факты описывают основную информацию о задаче.

```
1  node(a).
2  node(b).
3  node(c).
4  arc(a,b).
5  arc(b,c).
6
7  book('Title', '2009', 'Spb',
8    authors('First author', 'Second author')).
9
10 tree(tree(a,tree(b,c)),tree(d,e)).
```

Правила в Прологе

Правила (Вывод :- Условие): Вывод ИСТИНА, если Условие ИСТИНА.

```
1  % Equal to node(x).
2  node(x) :- true.
3
4  %and
5  doubleArc(X,Z) :- arc(X,Y), arc(Y,Z).
6
7  %or
8  edge(X,Y) :- arc(X,Y).
9  edge(X,Y) :- arc(Y,X).
```

Запросы в Прологе - 1

```
?- node(a).
true.

?- node(x).
false.

?- node(X).
X = a ;
X = b.

?- arc(a,Y), arc(Y,Z).
Y = b,
Z = c.
```

Запросы в Прологе - 2

```
?- X is 2+2.
X = 4.

?- A = 2 + 3 * 5, B is A.
A = 2+3*5,
B = 17.

?- List = [a, b, [c, d], e].
List = [a, b, [c, d], e].

?- [a,b,c,d] = [H | T].
H = a,
T = [b, c, d].
```

Основные механизмы работы

Унификация(unification)

Подстановка значений вместо переменных при выводе логической формулы.
Используется для выбора подходящего правила и для получения ответа.

Поиск с возвратом(backtracking)

Сводится к последовательному расширению частичного решения. Если на очередном шаге такое расширение провести не удается, то возвращаются к более короткому частичному решению и продолжают поиск дальше.
Используется для проверки альтернатив.

Пример решения ребуса

TWO + TWO = FOUR

```
1  sol(X) :-
2    =(X, [F, T, U, W, R, O]),
3    domain([F, T], 1, 9),
4    domain([U, W, R, O], 0, 9),
5              100*T + 10*W + O +
6              100*T + 10*W + O =:=
7    1000*F + 100*O + 10*U + R,
8    all_different(X).
9
10 % Definition of functions
11 % domain and all_different
12
13 ?- sol(X).
14 X = [1, 7, 3, 6, 0, 5] .
```

Удовлетворение ограничениям на Прологе

В чем проблема?

```
?- X=1+2
X = 1+2;
```

Хотелось бы перейти от синтаксического равенства термов к удовлетворению ограничениям.

```
?- domain([X,Y],0,100), 3#=X+Y, Y#>=2, X#>=1.
```

Для каждой переменной определяется домен — набор допустимых значений переменных (только дискретные с конечной областью определения, которые можно сопоставить с integer)

Реализация ограничений

- Для каждой переменной хранится текущий домен
- При добавлении ограничения, несовместимые значения удаляются из домена

Пример

	X	**Y**
	inf..*sup*	*inf*..*sup*
$domain([X, Y], 0, 100)$	0..100	0..100
$3 \#= X + Y$	0..3	0..3
$Y \#>= 2$	0..1	2..3
$X \#>= 1$	1	2

Пример решения ребуса с использованием clpfd

SEND + MORE = MONEY

```
1  % compare 6.8 s to 0 s!
2  :- use_module(library(clpfd)).
3
4  solve(Sol):-
5    Sol = [S,E,N,D,M,O,R,Y],
6    [E,N,D,O,R,Y] ins 0..9,
7    [S,M] ins 1..9,
8                1000*S + 100*E + 10*N + D +
9                1000*M + 100*O + 10*R + E #=
10   10000*M + 1000*O + 100*N + 10*E + Y,
11   all_different([S,E,N,D,M,O,R,Y]),
12   labeling([], Sol).
13
14 ?- solve(X).
15 X = [9, 5, 6, 7, 1, 0, 8, 2] ;
16 false.
```

Перебор с возвратом и фильтрация

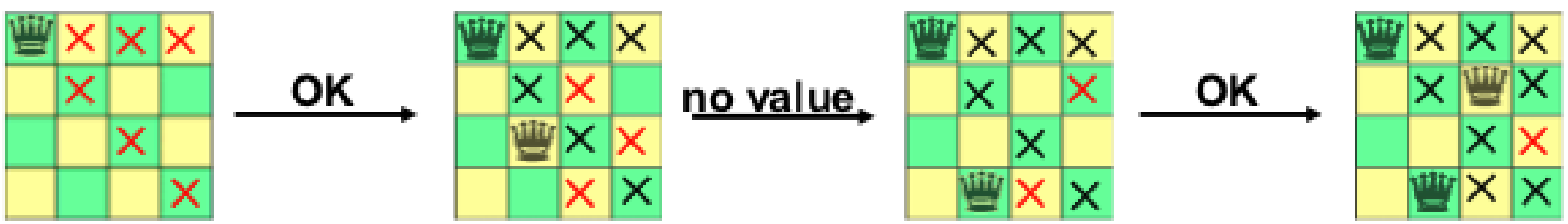
Почему перебор с возвратом настолько хуже фильтрации по домену?

Перебор с возвратом

Обнаруживает проблему поздно, рассматривает больше вариантов

Фильтрация по домену

Пересматривая ограничения, может отбросить неправильные варианты еще до их проверки



Еще один пример

```
1  :- use_module(library(clpfd)).
2
3  n_factorial(0, 1).
4  n_factorial(N, F) :-
5    N #> 0,
6    N1 #= N - 1,
7    F #= N * F1,
8    n_factorial(N1, F1).
9
10 ?- n_factorial(47, F).
11 F = 258623241511168180642964355153
12 611979969197632389120000000000 ;
13
14 ?- n_factorial(N, 24).
15 N = 4 ;
```

Вопросы

Ответы

```
% library(swi_hooks) compiled into pce_swi_hooks 0.00 sec, 3,856 bytes
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 5.10.4)
Copyright (c) 1990-2011 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?- Sol
|    .
% ... 1,000,000 ............ 10,000,000 years later
%
%       >> 42 << (last release gives the question)
?-
```

Литература

- Стюарт Рассел, Питер Норвиг, Искусственный интеллект: современный подход
- `http://www.swi-prolog.org/man/clpfd.html`
- Лекции R. Bartak "Язык Пролог и программирование в ограничениях"
- `http://ru.wikipedia.org/wiki/Prolog`