Problemy więzowe

Paweł Rychlikowski

Instytut Informatyki UWr

18 marca 2021

ale najpierw jeszcze trochę o A*

Heurystyki niedopuszczalne

- Heurystyki mogą być niedopuszczalne (w szczególności, jeżeli są wynikiem uczenia się heurystyk)
- Oczywiście tracimy wówczas (w teorii i praktyce) gwarancje optymalności.
- Ale można otrzymać istotnie szybsze wyszukiwanie (o czym, mam nadzieję, przekonamy się na pracowni 2)

Heurystyki niedopuszczalne w praktyce

- **Pytanie**: Jaka jest najprostsza heurystyka niedopuszczalna (nieoptymistyczna)?
- **Odpowiedź**: $(1+\varepsilon)h(n)$, gdzie h jest dopuszczalna

Czy ma ona jakiś sens?

Heurystyki niedopuszczalne w praktyce

- **Pytanie**: Jaka jest najprostsza heurystyka niedopuszczalna (nieoptymistyczna)?
- **Odpowiedź**: $(1+\varepsilon)h(n)$, gdzie h jest dopuszczalna

Czy ma ona jakiś sens?

Dla małego ϵ będziemy rozstrzygać remisy oryginalnej funkcji f preferując węzły, które wydają się być bliższe celowi.

Algorytm A*. Podsumowanie

Algorytm

Przeprowadź przeszukanie, wykorzystując $f(n)$ jako priorytet węzła (czyli rozwijamy węzły od tego, który ma najmniejszy f).

Kluczowa właściwość

1. A^* rozwija wszystkie węzły t.że $f(n) < C^*$.
2. A^* rozwija niektóre węzły t.że $f(n) = C^*$
3. A^* nie rozwija węzłów t.że $f(n) > C^*$.

Problemy więzowe

Problem spełnialności więzów

Uwaga

Między ósemką a hetmanami jest istotna różnica (mimo, że oba można przedstawiać jako problemy przeszukiwania).

- W ósemce interesuje nast droga dotarcia do celu, który jest dobrze znany (i tym samym mało ciekawy)
- W hetmanach interesuje nas, jak wygląda cel – droga do niego może być dość trywialna (dostawianie po kolei poprawnych hetmanów, przestawianie hetmanów z losowego ustawienia).

Problem spełnialności więzów (2)

Problemy takie jak hetmany są:

a) Bardzo istotne (ze względu na ich występowanie w rzeczywistym świecie)

b) Na tyle specyficzne, że warto dla nich rozważać specjalne metody.

Problemy spełnialności więzów. Definicja

Definicja

Problem spełnialności więzów ma 3 komponenty:

1. Zbiór zmiennych $X_1, \ldots, X_n$
2. Zbiór dziedzin (przypisanych zmiennym)
3. Zbiór więzów, opisujących dozwolone kombinacje wartości jakie mogą przyjmować zmienne.

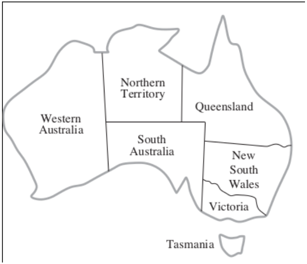
Przykład

Zmienne: X, Y, Z
Dziedziny: $X \in \{1, 2, 3, 4\}, Y \in \{1, 2\}, Z \in \{4, 5, 6, 7\}$
Więzy: $X + Y \geq Z, X \neq Y$

Komentarz do definicji

- Powyższy przykład to były więzy na dziedzinach skończonych, jeden z najważniejszych przypadków więzów.
- Ale można rozważać inne dziedziny:
  a) liczby naturalne,(trochę boli, że to nierozstrzygalny problem)
  b) liczby wymierne,
  c) ciągi elementów, napisy
  d) krotki
- Więzy określają relacje, często da się je wyrazić wzorem, ale nie jest to wymagane.

Kolorowanie Australii



- Mamy pokolorować mapę Australii, za pomocą 3 kolorów: (R, G, B)
- Sąsiadujące prowincje muszą mieć różne kolory.

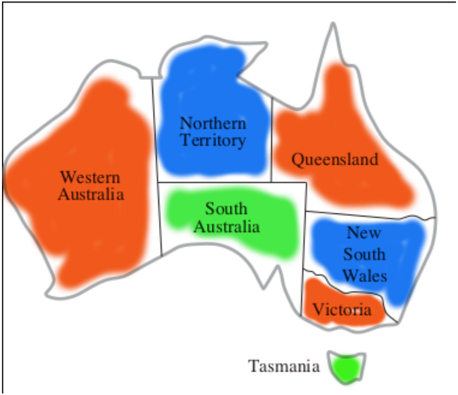
Kolorowanie jako problem więzowy

Zmienne: WA, NT, Q, NSW, V, SA, T

Dziedziny: {R,G,B}

Więzy: $SA \neq WA, SA \neq NT, SA \neq Q, SA \neq NSW, SA \neq V, WA \neq NT, NT \neq Q, Q \neq NSW, NSW \neq V$

Przykładowe kolorowanie

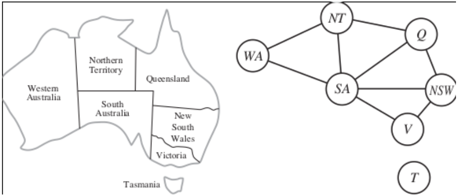


Arność więzów

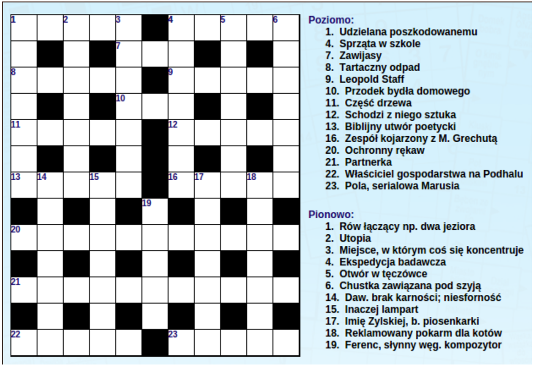
- Więzy (jako relacje) mogą mieć różną arność.
- Unarne – można potraktować jako modyfikację dziedziny (Tasmania nie jest czerwona) i zapomnieć.
- Binarne – jak w naszym przykładzie z kolorowaniem
- Mogą mieć też inną arność, w zasadzie dowolną (w praktyce spotyka się więzy o arności np. kilkaset)

Graf więzów

- Dla więzów binarnych możemy stworzyć graf, w którym krawędź oznacza, że dwie zmienne są powiązane więzem.
- Więzy binarne są istotną klasą więzów, wiele algorytmów działa przy założeniu binarności więzów.



Problemy dualne



Pytanie

Co powinno być zmienną w zadaniu rozwiązywania krzyżówki?

Krzyżówka (podstawowa)

Pomijamy (chwilowo?) kwestie zgodności hasła z definicją.

- Zmienne odpowiadają kratkom, dziedziną są znaki
- Więzy (fragment):
  jest-słowem-7(A,B,C,D,E,F,G), jest-słowem-3(B,H,I), ...

Krzyżówka (dualna)

- Zmienne to słowa (dziedziną jest słownik przycięty do określonej długości)
- Mamy więz dla każdej pary krzyżujących się słów. Jaki?

Więz dla słów

Przykładowy więz $C(w_1, w_2)$:

- w_1 ma długość 6
- w_2 ma długość 10
- trzeci znak w_1 jest taki sam, jak piąty znak w_2

Problemy dualne (2)

- Zwróćmy uwagę, że to, co zrobiliśmy z krzyżówką stosuje się do dowolnych więzów.
- Więzy w problemie prymarnym zmieniają się na zmienne w problemie dualnym (z dziedziną będącą dozwolonym zbiorem krotek)
- Dodatkowo potrzebujemu więzów, które mówią, że i-ty element jednej krotki jest j-tym elementem drugiej (te więzy są binarne!)

Koniec nagrania I

Propagacja więzów

Uwaga 1

To co odróżnia CSP od zadania przeszukiwania jest możliwość wykorzystania dodatkowej wiedzy o charakterze problemu do przeprowadzenia wnioskowania.

Uwaga 2

Podstawowym celem wnioskowania jest zmniejszenie rozmiaru dziedzin (a tym samym zmniejszenie przestrzeni przeszukiwań).

Wnioskowanie. Przykład

Przykład

Zmienne: X, Y, Z
Dziedziny: $X \in \{1, 2, 3, 4\}, Y \in \{1, 2\}, Z \in \{5, 6, 7, 8\}$
Więzy: $X + Y \geq Z, X \neq Y$

Czy możemy nie tracąc żadnego rozwiązania skreślić jakieś wartości z dziedzin?

Możemy wywnioskować, że: $X \in \{3, 4\}, Y \in \{1, 2\}, Z \in \{5, 6\}$

Jak widać, możemy też skreślić więz $X \neq Y$

Spójność więzów

- Spójność (intuicyjnie) rozumiemy jako niemożność wykreślenia żadnej wartości z dziedziny.
- Mamy różne rodzaje spójności:
  1. Węzłowa (każda wartość z dziedziny spełnia więzy unarne dla zmiennych)
  2. Łukowa: jak dwie zmienne są połączone więzem, to dla każdej wartości z dziedziny X jest wartość w dziedzinie Y, t.że dla tych wartości więz jest spełniony.

Uwaga

Są jeszcze inne rodzaje spójności. Więcej na ćwiczeniach.

Spójność więzów. Przykład

Więz: $X < Y$
Dziedzina X: $\{4, 6, 7, 10, 20\}$
Dziedzina Y: $\{1, 2, 4, 6, 7, 10\}$
Brak spójności

- Jeżeli weźmiemy X, to możemy wykreślić wartośći 10, 20
- Jeżeli weźmiemy Y, to możemy wykreślić wartości 1,2, 4

Po wykreśleniu tych wartości warto przyjrzeć się innym więzom z X i Y.

Spójność łukowa

Definicja

Więz C dla zmiennych X i Y z dziedzinami D_X i D_Y jest **spójny łukowo**, wtt:

- Dla każdego $x \in D_X$ istnieje takie $y \in D_Y$, że $C(x, y)$ jest spełnione
- Dla każdego $y \in D_Y$ istnieje takie $x \in D_X$, że $C(x, y)$ jest spełnione

Spójność łukowa

Definicja

Więz C dla zmiennych X i Y z dziedzinami D_X i D_Y jest **spójny łukowo**, wtt:

- Dla każdego $x \in D_X$ istnieje takie $y \in D_Y$, że $C(x, y)$ jest spełnione
- Dla każdego $y \in D_Y$ istnieje takie $x \in D_X$, że $C(x, y)$ jest spełnione

Sieć więzów jest spójna łukowo, jeżeli każdy więz jest spójny łukowo.

Algorytm AC-3

Algorytm zapewnia spójność łukową sieci więzów.

Idea

1. Zarządzamy kolejką więzów,
2. Usuwamy niepasujące wartości z dziedzin, analizując kolejne więzy z kolejki,
3. Po usunięciu wartości z dziedziny B, sprawdzamy wszystkie zmienne X, które występują w jednym więzie z B

```
function AC-3(csp) returns false if an inconsistency is found and true otherwise
   inputs: csp, a binary CSP with components (X, D, C)
   local variables: queue, a queue of arcs, initially all the arcs in csp

   while queue is not empty do
      (X_i, X_j) ← REMOVE-FIRST(queue)
      if REVISE(csp, X_i, X_j) then
         if size of D_i = 0 then return false
         for each X_k in X_i.NEIGHBORS - {X_j} do
            add (X_k, X_i) to queue
   return true
```

```
function REVISE(csp, X_i, X_j) returns true iff we revise the domain of X_i
   revised ← false
   for each x in D_i do
      if no value y in D_j allows (x,y) to satisfy the constraint between X_i and X_j then
         delete x from D_i
         revised ← true
   return revised
```

Algorytm AC-3. Uwagi

- Zwróćmy uwagę na niesymetryczność funkcji Revise (oznacza ona konieczność dodawania każdej pary zmiennych dwukrotnie)
- Zwróćmy uwagę, że istotny jest efekt uboczny tej funkcji: zmieniają się wartości dziedzin!

Złożoność algorytmu AC-3

- Mamy n zmiennych, dziedziny mają wielkość $O(d)$. Mamy c więzów.
- Obsługa więzu to $O(d^2)$
- Każdy więz może być włożony do kolejki co najwyżej $O(d)$ razy.

Złożoność

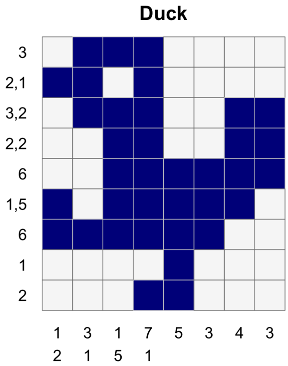
Złożoność wynosi zatem $O(cd^3)$ (raczej pesymistycznie)

Propagacja więzów

- Algorytm $AC-3$ jest przykładowym algorytmem **propagacji więzów**
- Przeprowadzamy rozumowanie, które pozwala nam **bezpiecznie** usuwać zmienne z dziedziny.
- Zmniejszanie dziedziny zmiennej X może spowodować zmniejszenie dziedzin innych, związanych z nią zmiennych.

Koniec części II

Spójność i obrazki logiczne



- Zmienne to wiersze i kolumny
- Spójność węzłowa (pojedynczy wiersz/kolumna zgodna ze specyfikacją)
- Spójność łukowa – zmiany w dziedzinie wierszu wpływają na kolumny i vice versa

Popatrzmy na przykład [inne okienko]

Hetmany

- (ubiegłoroczne) Zadanie z listy **Z1** pokazuje pewną technikę rozwiązywania zadań więzowych
- **Przypisuj wartości losowo i zarządzaj dziedzinami**

Wyjaśnienie w innym okienku

Uwaga

Bardziej systematyczny sposób nawiązujący do tej metody nazywa się **backtrackingiem** (przeszukiwaniem z nawrotami)

Poszukiwanie z nawrotami dla problemów więzowych

przeszukiwanie z nawrotami = backtracking search

- Wariant przeszukiwania w głąb, w którym stanem jest niepełne podstawienie.
- Nie pamiętamy całej historii, ale potrafimy zrobić **undo**
- Po każdym przypisaniu wykonujemy jakąś formę wnioskowania, bo może da się zmniejszyć dziedziny...

Backtracking

```
function BACKTRACK(assignment, csp) returns a solution, or failure
  if assignment is complete then return assignment
  var ← SELECT-UNASSIGNED-VARIABLE(csp)
  for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
     if value is consistent with assignment then
        add {var = value} to assignment
        inferences ← INFERENCE(csp, var, value)
        if inferences ≠ failure then
          add inferences to assignment
          result ← BACKTRACK(assignment, csp)
          if result ≠ failure then
             return result
     remove {var = value} and inferences from assignment
  return failure
```

Backtracking. Uwagi

1. Możliwy jest też taki wariant, że najpierw uruchamiamy AC-3, potem Backtracking z jakimś uproszczonym wnioskowaniem.
2. Wnioskowanie może nie tylko wykreślać elementy z dziedziny, może również dodawać inne więzy (implikacje)

W wielu sytuacjach, jak mechanizm wnioskowania jest silny, to wykonywane jest bardzo niewiele zgadnięć.

Parametry backtrackingu

1. Jak wybieramy zmienną do podstawienia (SelectUnassignedVariable)
2. W jakim porządku sprawdzamy dla niej wartości (OrderDomainValue)
3. Jak przeprowadzamy wnioskowanie (Inference)

Przykład. Plan lekcji

- Rozmieszczamy lekcje: zajęcia otrzymują termin
- Mamy naturalne więzy:
  - Jeżeli Z_1 i Z_2 mają tego samego nauczyciela (klasę, salę), wówczas $Z_1 \neq Z_2$
  - Nauczyciele nie mogą mieć zajęć o określonych porach (bo na przykład pracują w innych miejscach)
  - Wszystkie zajęcia klasy X danego dnia spełniają określone warunki: brak okienek, po jednej godzinie przedmiotu, itd.

Pytanie

W jakiej kolejności rozmieszcza zajęcia Pani Sekretarka?

Heurystyka: First Fail

Definicja

Wybieramy tę zmienną, która jest najtrudniejsza, co oznacza, że:

- ma najmniejszą dziedzinę,
- występuje w największej liczbie więzów.

Inne nazwy: Most Constrained First, Minimum Remaining Values (MRV)

Uzasadnienie

I tak będziemy musieli tę zmienną obsłużyć. Lepiej to zrobić, jak jeszcze inne zmienne są „wolne”

Wybór wartości

- Wybieramy tę wartość, która w najmniejszym stopniu ogranicza przyszłe wybory LCV, Least Contstraining Value.
- Przykład. W planie zajęć:
  1. Mamy teraz przydzielić termin zajęć panu A z klasą 1c
  2. Musimy później przydzielić zajęcia A z klasą 2a.
  3. Wcześniej przydzieliliśmy panią B z klasą 2a w czwartek na 8.
  4. Jest to argument za tym, żeby (A,1c) też była na ósmą w czwartek (bo nie stracimy żadnej możliwości dla (A,2a).

Wybór zmiennej vs wybór wartości

- W pierwszej chwili może dziwić przeciwne traktowanie wyboru zmiennych i wartości.
- Celem FirstFail jest agresywne ograniczanie przestrzeni poszukiwań.
- Celem LCV jest dążenie do jak najszybszego znalezienia pierwszego rozwiązania.

Musimy rozpatrzeć wszystkie zmienna, ale niekoniecznie wszystkie wartości!

Wybór zmiennej vs wybór wartości. Podsumowanie

- Wybieramy najgorszą zmienną
(ale każdą kiedyś musimy wybrać, a ta najgorsza najbardziej utrudni nam dalsze wybory)
- Wybieramy najlepszą wartość
(ale często zależy nam na znalezieniu pierwszego rozwiązania, nie wszystkich)

Uwaga

Możliwe inne heurystyki preferujące najbardziej obiecujące wartości! (można myśleć, że w tu jest miejsce na dowolny sensowny zachłanny algorytm wybierający wartość)