Iimplementazione del’algoritmo MAC (maintaining arc consistency) per problemi di sodisfacimento di vincoli(csp), utilizzando la tecnica AC3

Fatemah Alhamdoosh

Rappresentazione di un problema csp:

In python si può rappresentare un problema csp in una classe che ha degli attribuite:

variables: una lista di variabile.

Domain: un dizionario in cui ogni chiave è una variabile e il suo corrisponedente valore è una lista che rappresenta il domino della variabile.

Nieghbors: un dizionario in cui ogni chiave è una variabile e il suo corrisponedente valore è una lista di variabile vicini su cui si applica un vincolo.

Ha dei metodi assign e unassign per assegnare un valore e cancellare un valore della soluzione .

Il metodo IsConsistente (self, var, val, assignment), viene usato nel algoritmo Backtracing per decidere se un certo valore(val) del domino del variabile (val) in un assignment è consistente o meno.

La classe csp è implementato nel file **GenircClassCSP.py**

Algoritmo MAC maintaining arc consistency:

In python la funziona MACSearch prende come parametro una problema csp e ritorna una soluzione o None .

L'algoritmo è implementato nel file **MAC.py**

Test of Map coloring e Nqueens:

Ogni problema concreto si rappresenta come una classe che deriva la classe csp e assegna i variabili e i domini e i vicini relativi.

Inoltre definisce la funziona Constraint(self,A,a,B,b) che descrive il vincolo tra due variabile.

Colorare la mappa di australia:

mapColoringVariables=["SA","WA","NT", "Q" ,"NSW","V","T"]

neighbors={ "SA":["WA","NT","Q","NSW","V"],

"NT": ["WA","Q","SA"],

"NSW":["Q","V","SA"],

"WA":["NT","SA"],

"Q":["NT","SA","NSW"],

"V":["NSW","SA"] ,

"T": "" }

Tutti variabili hanno lo stesso dominio *["R","G","B"]*.

E alla fine il vincolo è che ogni due variabili vicini devono avere un colore diverso.

def constraint(self,A, a, B, b):

return a != b

NQueen:

Prende un numero n come parametro, e quindi i variabile sono una lista da 0 a n-1, *variabels= range(n)*

Tutti variabili hanno lo stesso dominio da 0 a n-1, e ogni variabile prende una lista di vicini da 0 a n-1.

Il vincolo in questa problema è definito in questo modo:

def constraint(self,A, a, B, b):

return A == B or (a != b and A + a != B + b and A - a != B - b)

Tutti dettagli descritti nel fil **InstanceOfProblemCSP.py**

Risultati:

Avviare il file **RunTest.py** fornisce tali risultati che dimostrano l'efficenza e la completezza del'lgoritmo maintain arc consistency.

Soluzione di map coloring:

{'WA': 'R', 'Q': 'R', 'T': 'R', 'V': 'R', 'SA': 'G', 'NT': 'B', 'NSW': 'B'}

Soluzione di Nqueen per n =8 (si può cambiare n al file RunTest.py):

Q . * . * . * .

. * . * . * Q *

* . * . Q . * .

. * . * . * . Q

* Q * . * . * .

. * . Q . * . *

* . * . * Q * .

. * Q * . * . *

Fonti :

https://solarianprogrammer.com/2017/11/20/eight-queens-puzzle-python/

http://aima.cs.berkeley.edu/python/csp.html