UNIVERSITÀ
DEGLI STUDI
FIRENZE

Scuola di Ingegneria

Corso di Laurea triennale in Ingegneria Informatica

Intelligenza Artificiale

Cutset conditioning

Febbraio 2020

Maggi Kevin - 6128567

INTRODUZIONE

L'elaborato consiste nell'implementazione dell'algoritmo di *Cutset Conditioning* per la risoluzione di CSP e relativo testing applicato al problema di *map-coloring* su mappe generate casualmente. I test eseguiti sono volti a mettere in luce il fatto che nel Cutset Conditioning devono essere scelte delle *euristiche* appropriate affinché si abbia un effettivo guadagno rispetto all'algoritmo di *Backtracking*.

L'implementazione degli algoritmi si basa su quelli visti a lezione e quelli presentati nel Russell-Norvig e nel Dechter.

IMPLEMENTAZIONE

Segue una breve panoramica su alcuni dettagli implementativi di rilievo sia per le strutture dati sia per gli algoritmi.

VARIABILE

La classe **Variable** rappresenta una variabile; consiste di soli due attributi: un set che rappresenta il suo dominio e un set che contiene gli eventuali valori nascosti del dominio in seguito ad inferenza (solamente quando essa risulta essere "definitiva" come in AC-3; nell'algoritmo MAC invece la propagazione di un vincolo può essere annullata dal meccanismo di backtracking, quindi verrà rappresentata in un altro modo illustrato successivamente).

VINCOLO

La classe callable **Constraint** rappresenta un vincolo binario; consiste di una funzione, che riceva due parametri e che basilarmente restituisca True/False quando due valori rispettano o meno il vincolo rappresentato, e un booleano *dual*. In generale i vincoli non sono commutativi (basti pensare alla relazione "maggiore di"), pertanto i due parametri passati alla chiamata a funzione devono essere ordinati; nella rappresentazione di un CSP è necessario rappresentare per ogni vincolo anche il duale, per evitare di dover definire due funzioni l'una inversa dell'altra per rappresentare entrambi i vincoli, l'attributo *dual* quando è settato a True indica che dovranno essere scambiati i due valori passati come parametri.

Vengono forniti anche una serie di funzioni di base (di comparazione) per la costruzione di vincoli.

CSP

La classe **CSP** consiste in un set di variabili e di due dizionari: uno per i vincoli binari e uno per i vincoli unari (che sono rappresentati come vincoli binari tra una variabile e un valore). Per semplicità si è assunto che tra due variabili o tra una variabile e un valore possa esserci un solo vincolo. Quando viene aggiunto un vincolo binario, viene automaticamente aggiunto anche il duale.

ASSEGNAMENTO

La classe **Assignment** rappresenta un assegnamento (parziale o totale); consiste di due dizionari: Il primo contiene l'assegnamento vero e proprio, ovvero a ogni variabile si fa corrispondere un valore assegnato, mentre nel secondo sono contenuti i valori "eliminati" dal dominio in seguito a inferenze che potrebbero non essere definitive (come in MAC: il fatto di non usare la funzione di inferenza inclusa nella classe Variable, ci permette di non dover tenere conto delle inferenze fatte, per poi poterle annullare in caso di backtracking. Infatti è sufficiente lavorare su una copia dell'assegnamento e in caso di backtracking ignorarla e tornare a lavorare su quella precedente).

AC-3

Questo algoritmo è stato implementato esattamente come visto a lezione. Le inferenze fatte vengono memorizzate direttamente nella variabile perché sono sicuramente definitive. Esegue in $O(nd^3)$.

BACKTRACK

Per questo algoritmo sono state usate le seguenti euristiche:

- *Minimum Remaining Value* e *Degree Heuristic* per la selezione della variabile da assegnare;
- *Mantaining Arc Consistency* per la propagazione dei vincoli (sostanzialmente AC-3 con in input l'insieme dei vicini della variabile assegnata)

Esegue in $O(d^n)$; spesso molto inferiore perché la *early termination* permette di fermarlo alla prima soluzione trovata, non ci interessano tutte.

TREE SOLVER

Questo algoritmo è stato implementato esattamente come visto a lezione. Esegue in $O(nd^2)$ e non lineare perché è limitato inferiormente dalla chiamata a *revise* contenuta in *Directional Arc Consistency*.

CUTSET

Questo algoritmo è stato implementato affinché esegua il Backtrack fintanto che il sottoproblema rimanente non è un albero, momento in cui passa al Tree Solver. Per fare ciò memorizza una copia semplificata del CSP (rappresentata dalla classe CSPWorkingCopy) e ad ogni iterazione controlla se il problema rimanente è un albero oppure no.

Per la particolare istanza di test che viene eseguita, può prendere un parametro "heuristic" che se settato a False (di default è True) sceglie l'ordine delle variabili in maniera casuale anziché secondo le euristiche MRV+DH.

La funzione *isATree* esegue un algoritmo simil-DFS.

MAP GENERATION

L'algoritmo di generazione casuale delle mappe è stato ripreso dall'esercizio 6.10 di R&N 2010: una volta generato il grafo della mappa, per renderlo un quasi-albero, ne viene trovato un albero ricoprente e in seguito scelti un numero di nodi (connessi tra loro) su cui aggiungere degli archi. Questi nodi rappresentano il *minimal cutset*, anche se poi non è garantito che l'algoritmo Cutset Conditioning lo identifichi (sia perché dipende dalle euristiche usate, ma soprattutto perché è un problema NP-Hard)

TEST

Vengono eseguiti i seguenti test:

- Backtracking – Cutset Conditioning con euristiche – Cutset Conditioning senza euristiche su mappe con minimal cutset di dimensione 1
- Backtracking – Cutset Conditioning con euristiche – Cutset Conditioning senza euristiche su mappe con minimal cutset di dimensione 2

E vengono presi in considerazione il tempo di esecuzione degli algoritmi e la dimensione del cutset sfruttato dalle due istanze dell'algoritmo cutset.

I test vengono eseguiti su mappe di dimensione (intesa come numero di regioni) crescente a multipli di 25 fino ad un massimo di 500. I tempi impiegati per risolvere questi CSP sono ancora molto bassi, ma la creazione delle mappe è molto onerosa in termini temporali.

RISULTATI

Sulle mappe con un minimal cutset costituito da una sola variabile il Cutset Conditioning con euristiche è quasi sempre riuscito ad individuare immediatamente il cutset, che è risultato essere quello minimo. È risultato quindi più efficacie in termini temporali del Backtracking.

Invece il Cutset Conditioning senza euristiche è risultato inefficiente, perché ha trovato un cutset di dimensioni prossime a quelle dell'intero problema; ovvero ha switchato al Tree Solver soltanto quando mancavano pochissime variabili. Questo si traduce in un tempo di esecuzione maggiore del Backtracking perché l'overhead costituito dal controllo (ogni iterazione) *isATree* ha superato (e non poco) il risparmio ottenuto grazie al Tree Solver.

Si sono verificati anche due casi in cui il cutset conditioning con euristica non ha rilevato immediatamente il minimal cutset (corrispondenti ai picchi nei grafici in figura 1) e in questo caso il suo tempo di esecuzione è stato prossimo a quello del cutset conditioning senza euristica.

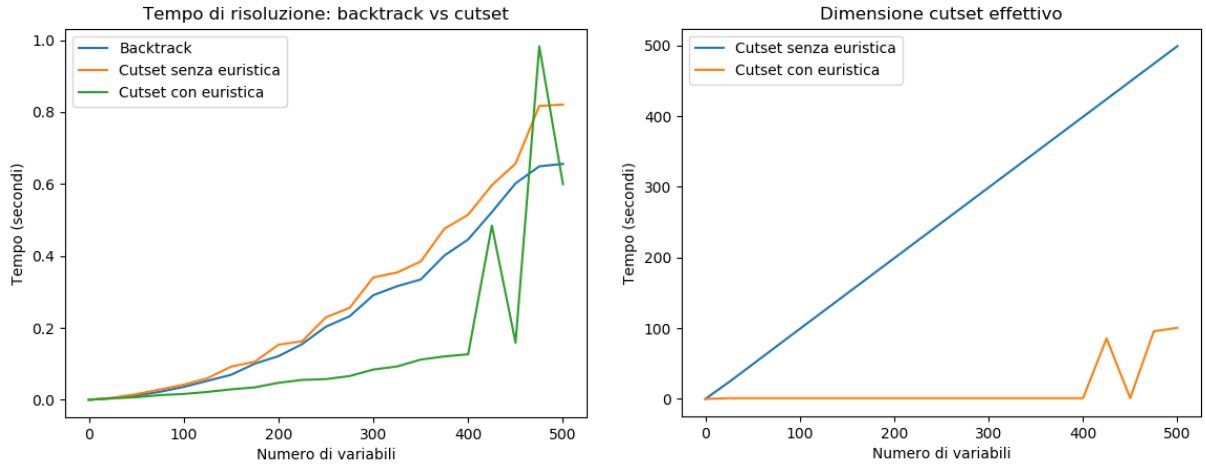

Figura 1 – Risultati su mappe con minimal cutset di dimensione 1

Sulle mappe con un minimal cutset costituito da 2 variabili anche il Cutset Conditioning con euristiche è risultato inefficiente. Ne consegue che il tempo minore di esecuzione è stato quello del Backtracking, che non presenta l'overhead per il controllo *isATree*. Si può vedere nei grafici in figura 2 che la dimensione media del cutset trovato è stata molto alta.

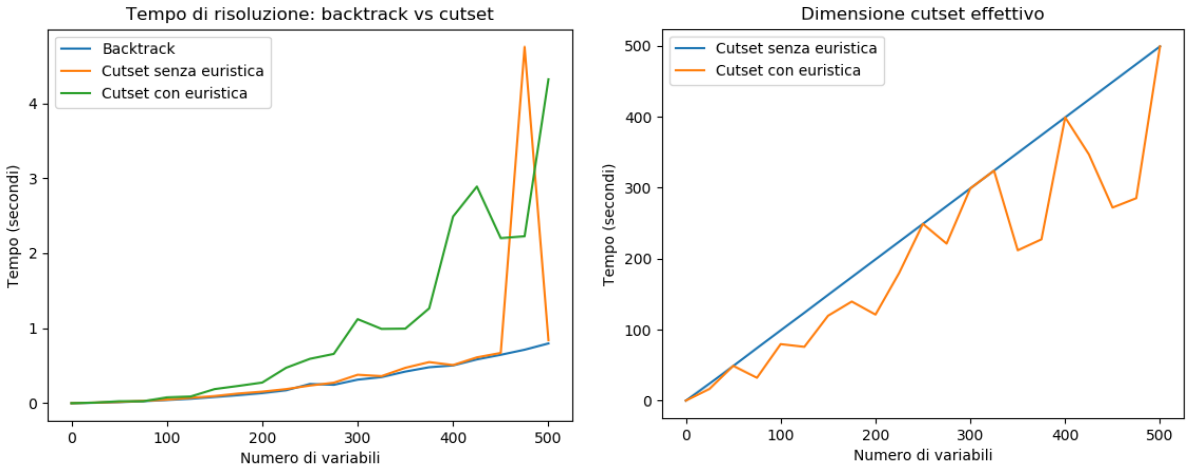

Figura 2 – Risultati su mappe con minimal cutset di dimensione 2

CONCLUSIONI

Alla luce dei test concludiamo che l'algoritmo di Cutset Conditioning offre dei vantaggi solamente nel caso in cui utilizzi delle euristiche appropriate. Nel nostro test le euristiche MRV e DH sono risultate utili nel primo caso, ma solamente perché per costruzione i minimal cutset dei problemi usati come test combaciavano con l'euristica stessa (il minimal cutset era, per come vengono costruite le mappe, quasi sempre la variabile con più nodi, identificata da DH).

Già nel secondo caso l'euristica non era più in grado di trovare il cutset in maniera efficiente (perché, appunto, non combaciavano più).

Su problemi di cui si ignora la natura del minimal cutset non è scontato che il Cutset Conditioning offra dei vantaggi; certo è che dovranno essere usate delle euristiche più sofisticate o almeno più informate.