Elaborato per esame finale AI

Gianluca Giuliani
Matricola 7003699

August 29, 2022

1 Introduzione

L'obiettivo dell'elaborato assegnatomi consiste nello sviluppo di un programma per la risoluzione di problemi di soddisfacimento dei vincoli utilizzando la tecnica del cutset-conditioning. Il programma viene verificato su due istanze di map-coloring e sudoku. Il linguaggio scelto per scrivere il programma è Java.

2 CutSet Conditioning

La tecnica si basa sull'ottenere un nearly-tree graph tramite la ricerca di certi nodi, detti cycleset, che creano cicli all'interno del constraint graph. Senza tale sottoinsieme di nodi, il grafo ha una struttura ad albero, senza cicli. L'algoritmo si basa sulla ricerca di questi nodi. Dopodiché trova un assegnamento completo che soddisfi tutti i vincoli, quindi effettua il topological sort sull'albero. Terminato il topological sort, si procede all'assegnazione dei nodi dell'albero.

3 Riproducibilità dei risultati

Se si desiderano riprodurre i risultati presenti su questo programma è sufficiente lanciare il programma. Se invece l'intenzione consiste nell'utilizzare il programma per risolvere nuove istanze allora è necessario prima scrivere le nuove variabili desiderate e inserire i vincoli tra di essi inserendo gli appositi comandi.

Si mostra tramite un esempio le operazioni da eseguire nel dettaglio. Si vuole risolvere un problema di Map Coloring diverso da quelli già inseriti.

3.1 Selezionare il dominio

Carico in un ArrayList di String (in questo caso) 3 colori diversi.

3.2 Istanziare un nuovo csp

Usando l'apposito costruttore (quello di Map Coloring) e passandogli come parametro l'ArrayList precedente.

3.3 Inserire le Variabili

Tramite il metodo addNode() aggiungo le variabili di mio interesse NOTA: si ricorda che il primo nodo inserito sarà il numero 0 nell'apposito attributo dell'istanza di Map Coloring, il secondo elemento

inserito sarà il numero 1 ecc...

3.4 Inserire i nodi

L'operazione viene svolta tramite addEdge(). Questo metodo richiede due parametri interi, quindi si deve fornire come parametro il numero intero con cui la variabile è memorizzata nell'attributo alist. E.g. si vuole inserire un arco tra la seconda variabile immessa al punto precedente e la terza -¿ i parametri da fornire saranno 1 e 2 rispettivamente, si faccia riferimento alla Nota soprastante.

3.5 Si noti

Si noti che nel caso di Sudoku, diventa tutto molto più semplice: Se si desidera inserire nella prima cella in alto a sinistra un certo valore è sufficiente usare il metodo setCellAssignment() che richiede tre parametri interi: Ordinata, ascissa e valore da inserire. Non si deve quindi ne' creare la variabile ne' creare il vincolo. Si noti che il conteggio delle righe e delle colonne parte da 0 e va dall'alto al basso e da sinistra verso destra rispettivamente. Si mostra, per completezza un esempio anche di questo caso: Si vuole inserire nella riga 0, colonna 5 il valore 6 allora è sufficiente scrivere: setCellAssignment(0,5,6).

3.6 Propagare gli input forniti

Si invoca il metodo controlAllElements(), permette di controllare se ci sono variabili critiche, cioè che posso assumere un solo valore, e assegna questo unico valore, o nessuno e si lancia il ripristino.

3.7 Cercare i nodi di Cycleset

Si chiama il metodo cercaNodo() che permette la rilevazione dei nodi appartenenti al cycleset: L'idea consiste nel trovare nodi che generano cicli nel grafo. Per farlo viene fatta una esplorazione. Entrano in gioco tre attributi dei nodi cycleset, explored, visiting. Questi booleani indicano, se veri, che il nodo è un Cycleset, che è stato usato come root dell'esplorazione, che è stato esplorato ma non era la root. L'idea molto intuitiva si basa sul fatto che scelto un nodo come root (e il suo explored diventa true), si vanno a esplorare i nodi ad esso collegato (e il loro visiting diventa true). Se durante l'esplorazione viene trovato un nodo il cui valore di visiting è true e non è già stato usato come radice per un'esplorazione precedente(explored == false), allora è cycleset . Questo viene aggiunto a Graph.cycleset. Terminata l'esplorazione si ripete e viene scelta come root il prossimo nodo in alist.

3.8 Assegnamento dei valori nel Cycleset

Viene fatto un assegnamento a tutti i cycleset. Si deve permettere sia il rispetto dei vincoli tra le variabili del cycleset sia che i rimanenti nodi abbiano valori, validi, che possono assumere. Si noti che ad ogni assegnamento, si deve propagare che il valore assegnato non è più disponibile per i nodi ad esso collegati dai vincoli. Dopodiché si esaminano le conseguenze: Se l'attuale assegnamento comporta che un nodo (non ancora assegnato) non abbia valori disponibili validi allora la configurazione è sbagliata e si corregge tramite il ripristino dell'attuale assegnamento, optando, se possibile, per un valore diverso dal precedente.

3.9 Topological Sort

Si effettua il topological Sort tramite il metodo topologicalSort(). Si potrebbe scegliere un qualsiasi nodo ma qui per semplicità si sceglie sempre il primo valore in alist, non appartentenente al cycleset, disponibile. I nodi selezionati vengono aggiunti in Graph.topological.

3.10 Arc-Consistency

Esamino dall'ultimo nodo dell'albero fino al primo. Per ognuno di questi nodi osservo i suoi figli e controllo se per ogni valore che questi possono assumere esiste un valore valido del nodo padre che soddisfa i vincoli.

3.11 Assegnamento valori rimanenti

Si effettua l'ultimo assegnamento: quello dei nodi dell'albero rimanente, presenti nell'attributo topological, rispettando i vincoli.

4 Analisi risultati sperimentali

Quello che ci si aspetta è che l'algoritmo fornisca una soluzione valida in tempi brevi se questa esiste. Si noti che il problema non ha necessità di cercare la soluzione ottima ma bensì di trovarne una valida, cioè che permetta di fare un assegnamento completo e che soddisfi tutti i vincoli presenti tra le variabili. Se non esiste una soluzione valida, il programma elimina i dati presenti in alist e mostra a schermo che non esiste soluzione al problema.

5 Fonti

Per la realizzazione del progetto sono state utilizzate le seguenti fonti:
Appunti del corso di intelligenza artificiale,
Artificial Intelligence - A Modern Approach (3rd Edition). `https://en.m.wikipedia.org/wiki/Graph_coloring` Le istanze inserite ed usate per la verifica di questo codice sono le seguenti.
La prima istanza di Map Coloring è stata estratta dalla sezione 6.5 del testo citato, immagine 6.12.
La seconda istanza di Map Coloring si può trovre a questo link): `https://en.m.wikipedia.org/wiki/Graph_coloring`
Le istanze di Sudoku sono state ottenute da una applicazione (Sudoku su playStore) per Android.