Cutset Conditioning per CSP: implementazione e risultati

Samuel Bruno

Sommario

La relazione descrive l'implementazione di un solver per problemi di soddisfacimento di vincoli (CSP) basato sulla tecnica di *cutset conditioning*. All'interno è mostrato il metodo adottato, la struttura dei file sorgente, il flusso d'esecuzione del programma e i risultati sperimentali su istanze di *map coloring* e *cryptoarithmetic*. Vengono inoltre fornite indicazioni per riprodurre gli esperimenti.

1 Introduzione

Cutset conditioning è una tecnica che sfrutta la struttura del grafo dei vincoli: si sceglie un sottoinsieme di variabili, detto cutset, e si esplorano tutte le possibili assegnazioni per esse. Una volta fissato il cutset, le variabili rimanenti (detti residui) formano un grafo senza cicli, che può quindi essere risolto con algoritmi più semplici ed efficienti. Nel progetto, in particolare, il grafo residuo viene risolto tramite backtracking su alberi.

Per ogni assegnazione del cutset si costruisce e si risolve il problema residuo; se il residuo ammette una soluzione, questa viene combinata con i valori assegnati al cutset per ottenere una soluzione completa del CSP.

Ad esempio, nella mappa dell'Australia, la variabile South Australia (SA) appartiene a tutti i cicli. Se la si fissa, le altre regioni rimaste formano una struttura aciclica che può essere risolta tramite backtracking su alberi.

Questa strategia è discussa nella letteratura di riferimento (Russell & Norvig, Artificial Intelligence: A Modern Approach, §6.5.1; Dechter, Tractable Structures for Constraint Satisfaction Problems, §3.1).

2 Metodo utilizzato

In questa sezione vengono descritti i punti essenziali del metodo implementato:

1. **Rappresentazione CSP:** ogni istanza è rappresentata da:
   - `variables`: lista di nomi delle variabili;
   - `domains`: lista del dominio dei valori possibili;
   - `constraints`: lista di coppie (constraint_variables, constraint_function) dove la prima è una tupla di coppie di variabili (per le mappe ad esempio, sono le coppie di regioni adiacenti) mentre constraint_function è una funzione booleana che, dati i valori assegnati a quelle variabili, nello stesso ordine, restituisce `True` se il vincolo è soddisfatto.
2. **Selezione del cutset:** si costruisce il grafo di adiacenza considerando solo i vincoli binari; si applica un'euristica greedy che rimuove iterativamente la variabile di grado massimo finché non rimangono cicli (cycle-cutset).

3. **Combinazioni e verifica parziale:** si generano tutte le possibili combinazioni di valori per le variabili del cutset. Ogni assegnazione parziale viene controllata per verificare che rispetti i vincoli interni al cutset (ad esempio, che due regioni adiacenti non abbiano lo stesso colore). Se i vincoli non sono rispettati, quell'assegnazione viene scartata.

4. **Costruzione del problema residuo:** una volta fissato il cutset, il problema rimanente (che coinvolge tutte le altre variabili) diventa privo di cicli. A questo punto:

   - si mantengono i vincoli che riguardano solo le variabili residue;
   - i vincoli binari che collegano una variabile del cutset a una variabile residua vengono trasformati in vincoli unari sulla variabile residua, sfruttando i valori fissati del cutset;
   - eventuali vincoli più complessi che coinvolgono sia cutset che residuo verranno verificati solo quando l'assegnazione completa sarà disponibile.

5. **Risoluzione del residuo:** il problema residuo, ora ad albero, viene risolto con il backtracking per alberi. Se si trova una soluzione per il residuo, questa viene combinata con l'assegnazione del cutset per ottenere la soluzione completa del CSP. In caso contrario, si passa alla successiva combinazione di valori per il cutset.

3 Descrizione dei file sorgente e del loro scopo

Il progetto è organizzato in più file Python che realizzano l'implementazione della tecnica di cutset conditioning:

csp.py

Definisce la classe `CSP` che contiene variabili, domini e vincoli. Fornisce il metodo `is_consistent`, che serve a controllare passo dopo passo se una certa assegnazione parziale rispetta i vincoli già definiti. In questo modo, durante la ricerca di una soluzione, si scartano subito le strade che non possono portare a un risultato valido.

tree_solver.py

Implementa il metodo `tree_backtrack`: applica un backtracking ricorsivo, provando uno alla volta i valori nel dominio di ciascuna variabile e sfruttando is_consistent per scartare subito le combinazioni incompatibili. La funzione assume che il grafo dei vincoli sia già privo di cicli.

cutset.py

Qui si trovano le parti principali che implementano la tecnica del cutset conditioning. Nel dettaglio:

- una funzione costruisce il grafo di adiacenza delle variabili a partire dai vincoli binari, e utilizza una procedura euristica per individuare un insieme di variabili da rimuovere (il cutset) in modo da eliminare i cicli;
- Un'altra funzione si occupa di trasformare vincoli binari che collegano variabili del cutset e variabili residue in vincoli unari sulle variabili residue, fissando i valori già assegnati nel cutset;
- Infine, la funzione `solve_with_cutset` si occupa di: provare tutte le possibili assegnazioni per il cutset, di verificare quali siano ammissibili, di costruire di volta in volta il problema residuo e di risolverlo con tree_backtrack. Se il residuo ha una soluzione, questa viene combinata con i valori del cutset per ottenere una soluzione completa del CSP.

mapcolor.py e cryptarithmetic.py

Questi due file hanno lo scopo di generare istanze concrete del problema da dare in input al solver.

- `mapcolor.py` produce problemi di colorazione di mappe: la mappa dell'Australia (7 regioni e 3 colori), una mappa semplificata dell'Europa (11 regioni e 4 colori) e una piccola versione degli Stati Uniti (5 regioni e 3 colori). I vincoli stabiliscono che due regioni confinanti non possano avere lo stesso colore.
- `cryptarithmetic.py` produce puzzle di criptoaritmetica, ovvero `T+T=EE`, `SEND+MORE=MONEY` e `TWO+TWO+TWO=SIX`. In questo caso le lettere rappresentano cifre e i vincoli impongono sia che tutte le lettere abbiano valori diversi, sia che l'equazione sia rispettata.

main.py

E' lo script principale che gestisce l'esecuzione del programma. Per ogni istanza costruita (sia di mappe che di criptoaritmetica), lo script crea il corrispondente oggetto CSP, lo risolve tramite `solve_with_cutset` e salva l'intero risultato in un file di log all'interno della cartella `logs_of_istances`. I log sono utili per seguire i passaggi compiuti dal solver e per verificare i risultati ottenuti, anzichè farli stampare ad ogni iterazione in console.

4 Istanze testate e risultati

Elenco delle istanze eseguite e risultato osservato con la configurazione adottata:

Categoria	Istanza	Risultato
Map coloring	Australia (7 regioni, 3 colori)	Soluzione trovata
Map coloring	Europa semplificata (11 regioni, 4 colori)	Soluzione trovata
Map coloring	USA semplificata (5 regioni, 3 colori)	Soluzione trovata
Crypto	T + T = EE	Nessuna soluzione (problema mal posto)
Crypto	SEND + MORE = MONEY	Soluzione trovata
Crypto	TWO + TWO + TWO = SIX	Soluzione trovata

5 Esempio CSP: mappa Australia

Viene descritta la sequenza operativa eseguita dal programma sull'istanza `australia`.

Istanza Le variabili sono `['WA','NT','SA','Q','NSW','V','T']`, ciascuna con dominio $\{R, G, B\}$. I vincoli binari impongono che regioni confinanti abbiano colori diversi.

Passaggi principali del solver

1. **Costruzione del grafo dei vincoli.** Il modulo di cutset analizza i vincoli binari e costruisce il grafo di adiacenza tra le regioni. In questo grafo la regione SA risulta collegata a molte altre regioni, diventando quindi il nodo con grado più elevato.
2. **Individuazione del cutset.** L'algoritmo `find_cycle_cutset` seleziona SA come cutset. Rimuovendo questa variabile, il grafo delle regioni residue diventa privo di cicli significativi e quindi gestibile con metodi più semplici.
3. **Assegnazione delle variabili del cutset.** Si provano tutte le possibili assegnazioni di colore per SA. Ad esempio, fissando {`SA='R'`}, non si verificano violazioni tra i vincoli interni al cutset (che in questo caso è composto dalla sola variabile SA).

4. **Costruzione del problema residuo.** Le variabili rimanenti (quindi il residuo) sono {WA, NT, Q, NSW, V, T}. I vincoli che collegavano SA con queste regioni vengono trasformati in vincoli unari, ad esempio SA='R' implica WA≠'R', NT≠'R', e così via. I vincoli binari tra le regioni residue, come WA≠NT, Q≠NSW, ecc. restano invariati.

5. **Risoluzione del problema residuo.** Il solver ad albero (`tree_backtrack`) esplora i domini delle variabili residue rispettando i vincoli unari e binari. Se viene trovata una soluzione valida (soluzione parziale), questa viene unita all'assegnazione del cutset (ad esempio SA='R') per ottenere la soluzione completa della mappa.

6 Riproducibilità

Per riprodurre i risultati:

1. copiare tutti i file Python nella stessa directory;

2. E' preferibile avere installata l'ultima versione di Python 3 (non sono richieste librerie esterne);

3. eseguire `python3 main.py` (i file dei risultati sono creati in `logs_of_istances`).

Riferimenti

- S. Russell, P. Norvig — *Artificial Intelligence: A Modern Approach*, 4th ed. — sezione 6.5.1 (Cutset conditioning).

- R. Dechter — *Tractable Structures for Constraint Satisfaction Problems* (2006) — sezioni su cycle-cutset e w-cutset (§3.1).