Branch and bound

Risolviamo un problema di PLI per enumerazione implicita:

branch (ramificazione) partizioniamo la regione ammissibile in sottoregioni;

bound (valutazione) stimiamo il valore ottimo di ogni sottoproblema;

potatura scartiamo le sottoregioni che non contengono soluzioni migliori di quella corrente (chiudiamo nodi dell'albero decisionale);

visita stabiliamo un ordine in cui visitare i nodi dell'albero.

Branch

Per esempio branch binario con $P_1 = \{x \in \Omega \mid x_1 \leq \overline{x}_1\}$ e $P_2 = \{x \in \Omega \mid x_1 \geq \overline{x}_1\}$ con $\overline{x}$ soluzione del rilassamento continuo, o branch non binario con $P_i = \{x \in \Omega \mid x_1 = i\}$.

Bound

Abbiamo bisogno un metodo per determinare i bound dei sottoproblemi in maniera efficiente:

valutazione inferiore valore della funzione obiettivo per una qualunque soluzione ammissibile (per esempio l'arrotondamento della soluzione del rilassamento continuo);

valutazione superiore ottimo di un rilassamento:

- continuo;
- eliminazione di vincoli (e.g. cammino minimo con distanza massima percorribile: eliminiamo il vincolo di distanza ed otteniamo un problema facile da risolvere che fornisce un upper bound);
- somma di vincoli;
- ...

Potatura

Possiamo chiudere un nodo P_i se vale una di:

- P_i non ha soluzioni ammissibili
- $v_S(P_i) \leq v_I(P)$, cioè le soluzioni ammissibili di P_i sono peggiori della stima inferiore corrente della soluzione globale;
- $v_S(P_i) > v_I(P)$ e la soluzione ottima del rilassamento di P_i è ammissibile per P; aggiorniamo $v_I(P)$ a $v_S(P_i)$.

Se $v_S(P_i) > v_I(P)$ ma la soluzione ottima del rilassamento di P_i non è ammissibile per P, allora dobbiamo partizionare P_i.

Visita

depth first (profondità) trova rapidamente una soluzione ammissibile e occupa poca memoria, ma non tiene conto della qualità della soluzione trovata;

best first il prossimo nodo da visitare è quello con la massima valutazione superiore: trova rapidamente una soluzione ammissibile e tiene conto della qualità, ma occupa molta memoria;

breadth first (ampiezza) in generale cattive prestazioni..

Algoritmo

1. genera il nodo radice P, trova una sua soluzione ammissibile e $v_I(P)$;
2. se tutti i nodi sono stati visitati, stop – la soluzione corrente è ottima;
3. seleziona un nodo P_i da visitare;
4. se P_i non contiene soluzioni ammissibili, chiudilo e torna a 2;
5. bound: risolvi un rilassamento di P_i per calcolare $v_s(P_i)$
   - se $v_S(P_i) \leq v_I(P)$ chiudi P_i e torna a 2;
   - se $v_S(P_i) > v_I(P)$ e la soluzione ottima del rilassamento di P_i è ammissibile per P, chiudi P_i, aggiorna la soluzione ammissibile, poni $v_I(P) = v_S(P_i)$ e torna a 2;
6. branch: partiziona la regione ammissibile di P_i in sottoregioni e genera i nodi corrispondenti, poi torna a 2.

Inoltre, se abbiamo calcolato l'upper bound di tutti i figli di P_i possiamo aggiornare la sua valutazione superiore al massimo dei bound trovati.