Branch and Bound

Branch-and-bound

Tra le tecniche di risoluzione esatte per problemi difficili una molto popolare è quella denominata branch-and-bound.

Branch-and-bound

Tra le tecniche di risoluzione esatte per problemi difficili una molto popolare è quella denominata branch-and-bound.

Descriveremo l'algoritmo generico di branch-and-bound per problemi di massimo. Di seguito segnaleremo le piccole variazioni che vanno introdotte per problemi di minimo.

Branch-and-bound

Tra le tecniche di risoluzione esatte per problemi difficili una molto popolare è quella denominata branch-and-bound.

Descriveremo l'algoritmo generico di branch-and-bound per problemi di massimo. Di seguito segnaleremo le piccole variazioni che vanno introdotte per problemi di minimo.

Il problema di massimo considerato è

$$\max_{x \in S} f(x),$$

dove S è la regione ammissibile e f la funzione obiettivo del problema.

Componenti algoritmo:upper bound

Si consideri un sottinsieme $T \subseteq S$ della regione ammissibile. Una limitazione superiore o *upper bound* per T è un valore $U(T)$ con la seguente proprietà

$$U(T) \geq f(x) \quad \forall\, x \in T.$$

Il calcolo dell'upper bound

Il valore $U(T)$ viene calcolato tramite una procedura che deve cercare di soddisfare queste due proprietà in conflitto tra loro:

Il calcolo dell'upper bound

Il valore $U(T)$ viene calcolato tramite una procedura che deve cercare di soddisfare queste due proprietà in conflitto tra loro:

- i tempi di esecuzione della procedura devono essere brevi (in particolare, il calcolo degli upper bound deve richiedere un tempo molto inferiore rispetto al tempo necessario per risolvere l'intero problema);

Il calcolo dell'upper bound

Il valore $U(T)$ viene calcolato tramite una procedura che deve cercare di soddisfare queste due proprietà in conflitto tra loro:

- i tempi di esecuzione della procedura devono essere brevi (in particolare, il calcolo degli upper bound deve richiedere un tempo molto inferiore rispetto al tempo necessario per risolvere l'intero problema);
- il valore $U(T)$ deve essere il più vicino possibile al massimo valore di f su T.

Il calcolo dell'upper bound

Il valore $U(T)$ viene calcolato tramite una procedura che deve cercare di soddisfare queste due proprietà in conflitto tra loro:

- i tempi di esecuzione della procedura devono essere brevi (in particolare, il calcolo degli upper bound deve richiedere un tempo molto inferiore rispetto al tempo necessario per risolvere l'intero problema);
- il valore $U(T)$ deve essere il più vicino possibile al massimo valore di f su T.

Spesso la scelta di una procedura per il calcolo dell'upper bound è fortemente legata al particolare problema che si sta risolvendo. Inoltre, non esiste un'unica procedura per un dato problema.

Upper bound e rilassamento

Un modo comunemente utilizzato per determinare un upper bound $U(T)$ è quello di determinare la soluzione di un suo *rilassamento*.

Upper bound e rilassamento

Un modo comunemente utilizzato per determinare un upper bound $U(T)$ è quello di determinare la soluzione di un suo *rilassamento*.

Indichiamo con:

$$\alpha(f, T) = \max_{x \in T} f(x),$$

il valore ottimo della funzione f sull'insieme T.

Upper bound e rilassamento

Un modo comunemente utilizzato per determinare un upper bound $U(T)$ è quello di determinare la soluzione di un suo *rilassamento*.

Indichiamo con:

$$\alpha(f, T) = \max_{x \in T} f(x),$$

il valore ottimo della funzione f sull'insieme T.

Si definisce rilassamento del problema, un problema:

$$\alpha(f', T') = \max_{x \in T'} f'(x)$$

dove:

$$T \subseteq T' \quad \text{e} \quad f'(x) \geq f(x) \quad \forall \, x \in T.$$

Osservazione

Si ha che: $\alpha(f', T') \geq \alpha(f, T)$.

Osservazione

Si ha che: $\alpha(f', T') \geq \alpha(f, T)$.

Dimostrazione Sia $x^* \in T$ una soluzione ottima del problema su T, cioè:

$$f(x^*) = \alpha(f, T),$$

e sia $x' \in T'$ una soluzione ottima del rilassamento, cioè:

$$f'(x') = \alpha(f', T').$$

Osservazione

Si ha che: $\alpha(f', T') \geq \alpha(f, T)$.

Dimostrazione Sia $x^* \in T$ una soluzione ottima del problema su T, cioè:

$$f(x^*) = \alpha(f, T),$$

e sia $x' \in T'$ una soluzione ottima del rilassamento, cioè:

$$f'(x') = \alpha(f', T').$$

Si ha che $x^* \in T$ implica $x^* \in T'$.

Osservazione

Si ha che: $\alpha(f', T') \geq \alpha(f, T)$.

Dimostrazione Sia $x^* \in T$ una soluzione ottima del problema su T, cioè:

$$f(x^*) = \alpha(f, T),$$

e sia $x' \in T'$ una soluzione ottima del rilassamento, cioè:

$$f'(x') = \alpha(f', T').$$

Si ha che $x^* \in T$ implica $x^* \in T'$. Inoltre, si ha:

$$f'(x^*) \geq f(x^*).$$

Osservazione

Si ha che: $\alpha(f', T') \geq \alpha(f, T)$.

Dimostrazione Sia $x^* \in T$ una soluzione ottima del problema su T, cioè:

$$f(x^*) = \alpha(f, T),$$

e sia $x' \in T'$ una soluzione ottima del rilassamento, cioè:

$$f'(x') = \alpha(f', T').$$

Si ha che $x^* \in T$ implica $x^* \in T'$. Inoltre, si ha:

$$f'(x^*) \geq f(x^*).$$

Infine, l'ottimalità di x' implica $f'(x') \geq f'(x^*)$ e quindi:

$$\alpha(f', T') = f'(x') \geq f'(x^*) \geq f(x^*) = \alpha(f, T).$$

Lower bound

Un limite inferiore o *lower bound* per il valore ottimo del nostro problema è un valore LB che soddisfa la seguente proprietà:

$$LB \leq f(x^*) = \max_{x \in S} f(x).$$

Come si calcola?

Se prendiamo un qualsiasi elemento $\overline{x} \in S$ e valutiamo in esso la funzione f, il valore $f(\overline{x})$ è già un lower bound, dal momento che $f(\overline{x}) \leq f(x^*)$.

Come si calcola?

Se prendiamo un qualsiasi elemento $\overline{x} \in S$ e valutiamo in esso la funzione f, il valore $f(\overline{x})$ è già un lower bound, dal momento che $f(\overline{x}) \leq f(x^*)$.

Durante l'esecuzione di un algoritmo branch-and-bound la funzione f viene valutata per molti elementi $y_1, \ldots, y_h \in S$ e per ognuno di essi si ha

$$f(y_i) \leq f(x^*) \qquad i = 1, \ldots, h.$$

Come si calcola?

Se prendiamo un qualsiasi elemento $\overline{x} \in S$ e valutiamo in esso la funzione f, il valore $f(\overline{x})$ è già un lower bound, dal momento che $f(\overline{x}) \leq f(x^*)$.

Durante l'esecuzione di un algoritmo branch-and-bound la funzione f viene valutata per molti elementi $y_1, \ldots, y_h \in S$ e per ognuno di essi si ha

$$f(y_i) \leq f(x^*) \qquad i = 1, \ldots, h.$$

A noi interessa un valore LB il più possibile vicino al valore ottimo del problema. Quindi, poniamo

$$LB = \max\{f(y_i)\ :\ i = 1, \ldots, h\} \leq f(x^*).$$

Ma ...

... da dove ricaviamo gli elementi di S in cui valutare la funzione f durante l'esecuzione dell'algoritmo?

Ma ...

... da dove ricaviamo gli elementi di S in cui valutare la funzione f durante l'esecuzione dell'algoritmo?

- Se si ha a disposizione un'euristica è buona norma valutare f nel risultato di tale euristica;

Ma ...

... da dove ricaviamo gli elementi di S in cui valutare la funzione f durante l'esecuzione dell'algoritmo?

- Se si ha a disposizione un'euristica è buona norma valutare f nel risultato di tale euristica;
- durante lo stesso calcolo degli upper bound si possono individuare uno o più elementi di S e valutare in essi f.

Ma ...

... da dove ricaviamo gli elementi di S in cui valutare la funzione f durante l'esecuzione dell'algoritmo?

- Se si ha a disposizione un'euristica è buona norma valutare f nel risultato di tale euristica;
- durante lo stesso calcolo degli upper bound si possono individuare uno o più elementi di S e valutare in essi f. Ad esempio, se si calcola l'upper bound $U(T)$ tramite un rilassamento, nei casi in cui per la soluzione $x' \in T' \supseteq T$ valga anche $x' \in T$, allora si ha anche $x' \in S$ e si può valutare f in x'.

Ma ...

... da dove ricaviamo gli elementi di S in cui valutare la funzione f durante l'esecuzione dell'algoritmo?

- Se si ha a disposizione un'euristica è buona norma valutare f nel risultato di tale euristica;
- durante lo stesso calcolo degli upper bound si possono individuare uno o più elementi di S e valutare in essi f. Ad esempio, se si calcola l'upper bound $U(T)$ tramite un rilassamento, nei casi in cui per la soluzione $x' \in T' \supseteq T$ valga anche $x' \in T$, allora si ha anche $x' \in S$ e si può valutare f in x'. In altri casi non si ha $x' \in T$ ma con opportune operazioni (quali arrotondamenti o approssimazioni per eccesso/difetto di valori di variabili) si può determinare partendo da $x' \notin T$ una soluzione $\overline{x}' \in T$ (un esempio di ciò lo incontreremo nell'algoritmo branch-and-bound per il problema dello zaino).

Branching

L'operazione di branching consiste nel rimpiazzare un insieme $T \subseteq S$ con una sua partizione $T_1, \ldots, T_m$. Si ricordi che $T_1, \ldots, T_m$ formano una partizione di T se

$$T = \cup_{i=1}^{m} T_i \quad T_i \cap T_j = \emptyset \;\; \forall \; i \neq j.$$

Branching

L'operazione di branching consiste nel rimpiazzare un insieme $T \subseteq S$ con una sua partizione $T_1, \ldots, T_m$. Si ricordi che $T_1, \ldots, T_m$ formano una partizione di T se

$$T = \cup_{i=1}^{m} T_i \quad T_i \cap T_j = \emptyset \ \ \forall\, i \neq j.$$

La partizione può essere rappresentata tramite una struttura ad albero: l'insieme T è un nodo dell'albero da cui partono i rami (da qui il nome branching) verso i nodi della partizione, che vengono anche detti nodi successori o nodi figli del nodo T.

Cancellazione di sottinsiemi

Il punto chiave degli algoritmi di branch-and-bound è la *cancellazione di sottinsiemi*.

Cancellazione di sottinsiemi

Il punto chiave degli algoritmi di branch-and-bound è la *cancellazione di sottinsiemi*.

Supponiamo che per un dato sottinsieme, T_2 ad esempio, si abbia

$$U(T_2) \leq LB.$$

Cancellazione di sottinsiemi

Il punto chiave degli algoritmi di branch-and-bound è la *cancellazione di sottinsiemi*.

Supponiamo che per un dato sottinsieme, T_2 ad esempio, si abbia

$$U(T_2) \leq LB.$$

Ma questo vuol dire che

$$\forall\, x \in T_2 \quad f(x) \leq U(T_2) \leq LB,$$

e cioè tra tutti gli elementi in T_2 non ne possiamo trovare alcuno con valore di f superiore a LB, ovvero al miglior valore di f osservato fino a questo momento. A questo punto posso *cancellare* il sottinsieme T_2.

Cancellazione ed enumerazione implicit

La cancellazione equivale ad una *enumerazione implicita*: il confronto tra upper bound $U(T_2)$ del sottinsieme e lower bound LB ci consente di scartare tutti gli elementi in T_2 *senza dover calcolare la funzione* f *in essi*.

Cancellazione ed enumerazione implicit

La cancellazione equivale ad una *enumerazione implicita*: il confronto tra upper bound $U(T_2)$ del sottinsieme e lower bound LB ci consente di scartare tutti gli elementi in T_2 *senza dover calcolare la funzione f in essi*.

La regola di cancellazione appena introdotta ci fa capire perché vogliamo un valore di upper bound $U(T_2)$ il più vicino possibile al valore ottimo di f sul sottinsieme T_2 e un valore LB il più possibile vicino al valore ottimo del problema:

Cancellazione ed enumerazione implicit

La cancellazione equivale ad una *enumerazione implicita*: il confronto tra upper bound $U(T_2)$ del sottinsieme e lower bound LB ci consente di scartare tutti gli elementi in T_2 *senza dover calcolare la funzione f in essi*.

La regola di cancellazione appena introdotta ci fa capire perché vogliamo un valore di upper bound $U(T_2)$ il più vicino possibile al valore ottimo di f sul sottinsieme T_2 e un valore LB il più possibile vicino al valore ottimo del problema:

in questo modo è più semplice cancellare il sottinsieme tramite la condizione $U(T_2) \leq LB$.

L'algoritmo branch-and-bound

- **Passo 1** Si ponga $\mathcal{C} = \{S\}$ e $\mathcal{Q} = \emptyset$ (l'insieme $\mathcal{C}$ conterrà sempre i sottinsiemi ancora da tenere in considerazione e inizialmente contiene l'intero insieme S, mentre l'insieme $\mathcal{Q}$, inizialmente vuoto, conterrà tutti i sottinsiemi cancellati). Si ponga $k = 1$. Si calcoli $U(S)$ e si calcoli un valore per LB (eventualmente utilizzando anche i risultati di un'euristica, se disponibile). Se non si dispone di soluzioni ammissibili, si ponga $LB = -\infty$.

L'algoritmo branch-and-bound

- **Passo 1** Si ponga $\mathcal{C} = \{S\}$ e $\mathcal{Q} = \emptyset$ (l'insieme $\mathcal{C}$ conterrà sempre i sottinsiemi ancora da tenere in considerazione e inizialmente contiene l'intero insieme S, mentre l'insieme $\mathcal{Q}$, inizialmente vuoto, conterrà tutti i sottinsiemi cancellati). Si ponga $k = 1$. Si calcoli $U(S)$ e si calcoli un valore per LB (eventualmente utilizzando anche i risultati di un'euristica, se disponibile). Se non si dispone di soluzioni ammissibili, si ponga $LB = -\infty$.
- **Passo 2 (Selezione di un sottinsieme)** Si selezioni un sottinsieme $T \in \mathcal{C}$. Tra le varie regole di selezione citiamo qui quella di selezionare il sottinsieme T in $\mathcal{C}$ con il valore di upper bound più elevato, cioè

$$U(T) = \max_{Q \in \mathcal{C}} U(Q).$$

- **Passo 3 (Branching)** Si sostituisca l'insieme T in $\mathcal{C}$ con la sua partizione in m_k sottinsiemi $T_1, \ldots, T_{m_k}$, ovvero

$$\mathcal{C} = \mathcal{C} \cup \{T_1, \ldots, T_{m_k}\} \setminus \{T\}.$$

- **Passo 3 (Branching)** Si sostituisca l'insieme T in $\mathcal{C}$ con la sua partizione in m_k sottinsiemi $T_1, \ldots, T_{m_k}$, ovvero

$$\mathcal{C} = \mathcal{C} \cup \{T_1, \ldots, T_{m_k}\} \setminus \{T\}.$$

- **Passo 4 (Upper bounding)** Si calcoli un upper bound $U(T_i)$, $i = 1, \ldots, m_k$ per ogni sottinsieme della partizione.

- **Passo 3 (Branching)** Si sostituisca l'insieme T in $\mathcal{C}$ con la sua partizione in m_k sottinsiemi $T_1, \ldots, T_{m_k}$, ovvero

$$\mathcal{C} = \mathcal{C} \cup \{T_1, \ldots, T_{m_k}\} \setminus \{T\}.$$

- **Passo 4 (Upper bounding)** Si calcoli un upper bound $U(T_i)$, $i = 1, \ldots, m_k$ per ogni sottinsieme della partizione.
- **Passo 5 (Lower bounding)** Si aggiorni, eventualmente, il valore LB (si ricordi che il valore LB corrisponde sempre al massimo dei valori di f osservati durante l'esecuzione dell'algoritmo).

- **Passo 6 (Cancellazione sottinsiemi)** Si escludano da $\mathcal{C}$ tutti i sottinsiemi Q per cui $U(Q) \leq LB$, ovvero

$$\mathcal{C} = \mathcal{C} \setminus \{Q : \ U(Q) \leq LB\}.$$

e si trasferiscano tali sottinsiemi in $\mathcal{Q}$, cioè:

$$\mathcal{Q} = \mathcal{Q} \cup \{Q : \ U(Q) \leq LB\}.$$

- **Passo 6 (Cancellazione sottinsiemi)** Si escludano da $\mathcal{C}$ tutti i sottinsiemi Q per cui $U(Q) \leq LB$, ovvero

$$\mathcal{C} = \mathcal{C} \setminus \{Q : \; U(Q) \leq LB\}.$$

e si trasferiscano tali sottinsiemi in $\mathcal{Q}$, cioè:

$$\mathcal{Q} = \mathcal{Q} \cup \{Q : \; U(Q) \leq LB\}.$$

- **Passo 7** Se $\mathcal{C} = \emptyset$: stop, il valore LB coincide con il valore ottimo $f(x^*)$. Altrimenti si ponga $k = k + 1$ e si ritorni al Passo 2.

Osservazione

Se $\mathcal{C} = \emptyset$, LB è il valore ottimo del nostro problema (se è pari a $-\infty$, allora $S = \emptyset$).

Osservazione

Se $\mathcal{C} = \emptyset$, LB è il valore ottimo del nostro problema (se è pari a $-\infty$, allora $S = \emptyset$).

Questa affermazione è una conseguenza del fatto che, nel momento in cui $\mathcal{C} = \emptyset$, tutti i sottinsiemi cancellati fino a quel momento, cioè la collezione $\mathcal{Q}$ di sottinsiemi, formano una *partizione dell'intero insieme* S.

Osservazione

Se $\mathcal{C} = \emptyset$, LB è il valore ottimo del nostro problema (se è pari a $-\infty$, allora $S = \emptyset$).

Questa affermazione è una conseguenza del fatto che, nel momento in cui $\mathcal{C} = \emptyset$, tutti i sottinsiemi cancellati fino a quel momento, cioè la collezione $\mathcal{Q}$ di sottinsiemi, formano una *partizione dell'intero insieme* S.

Quindi tra di essi ve ne è certamente uno, indicato con $T^* \in \mathcal{Q}$, che contiene x^*. Ma poiché T^* è stato cancellato si dovrà avere

$$f(x^*) \leq U(T^*) \leq LB \leq f(x^*),$$

da cui segue immediatamente che $LB = f(x^*)$.

Modifiche per problemi di minimo

- ad un sottinsieme $Q \subseteq S$ dovrà essere associato un valore di lower bound $L(Q)$;

Modifiche per problemi di minimo

- ad un sottinsieme $Q \subseteq S$ dovrà essere associato un valore di lower bound $L(Q)$;
- al posto del valore LB avremo un valore UB con la proprietà
$$UB \geq f(x^*) = \min_{x \in S} f(x).$$
Il valore UB sarà il minimo tra i valori osservati della funzione obiettivo in punti della regione ammissibile S.

Modifiche per problemi di minimo

- ad un sottinsieme $Q \subseteq S$ dovrà essere associato un valore di lower bound $L(Q)$;
- al posto del valore LB avremo un valore UB con la proprietà
$$UB \geq f(x^*) = \min_{x \in S} f(x).$$
Il valore UB sarà il minimo tra i valori osservati della funzione obiettivo in punti della regione ammissibile S.
- Il sottinsieme Q viene cancellato se è vero che $L(Q) \geq UB$.

Modifiche per problemi di minimo

- ad un sottinsieme $Q \subseteq S$ dovrà essere associato un valore di lower bound $L(Q)$;
- al posto del valore LB avremo un valore UB con la proprietà
$$UB \geq f(x^*) = \min_{x \in S} f(x).$$
Il valore UB sarà il minimo tra i valori osservati della funzione obiettivo in punti della regione ammissibile S.
- Il sottinsieme Q viene cancellato se è vero che $L(Q) \geq UB$.
- Al Passo 2 della procedura di branch-and-bound si seleziona un nodo con lower bound più piccolo, ovvero un nodo T tale che
$$L(T) = \min_{Q \in \mathcal{C}} L(Q).$$

Nota bene

Come già osservato parlando della cancellazione di nodi, l'algoritmo branch-and-bound è un algoritmo di enumerazione implicita.

Nota bene

Come già osservato parlando della cancellazione di nodi, l'algoritmo branch-and-bound è un algoritmo di enumerazione implicita.

Invece di valutare esplicitamente il valore della funzione obiettivo in tutti i punti di un sottinsieme, lo si può escludere in blocco sulla base del valore del proprio upper bound (o lower bound nei problemi di minimo). Tale bound è calcolato attrverso la risoluzione di un opportuno sottoproblema (quello che abbiamo chiamato un rilassamento).