Algoritmi e Strutture di Dati (3^a Ed.)
Ricerca tabù

Alan Bertossi, Alberto Montresor

La tecnica della ricerca locale passa attraverso una sequenza S_0, S_1, …, S_m di soluzioni, fino ad arrestarsi su un ottimo locale S_m. Nel passaggio da una soluzione S_t alla soluzione successiva S_{t+1}, è applicata una certa regola di trasformazione. Un modo equivalente di descrivere tale procedimento corrisponde ad effettuare una "mossa", tra un certo insieme di mosse possibili, in modo da selezionare S_{t+1} nell'intorno di S_t. Denotato con $M = \{\mu_1, \ldots, \mu_p\}$ l'insieme delle mosse possibili, l'intorno $I(S_t)$ al passo t-esimo è definito come:

$$I(S_t) = \{S : S \text{ è ottenibile da } S_t \text{ applicando una mossa } \mu_i \in M\}.$$

In questo modo, l'intorno $I(S_t)$ dipende dall'ultima soluzione S_t e dall'intero insieme M delle mosse possibili. In generale, però, si può definire l'intorno in modo che dipenda dalla sequenza S_0, S_1, …, S_t di soluzioni e da un sottoinsieme M' di M. In altri termini, alcune delle soluzioni in $I(S_t)$ non possono essere selezionate perché le rispettive mosse μ_i sono proibite. Un semplice modo per proibire alcune mosse consiste nell'introduzione di un parametro T di proibizione, che determina per quanto tempo una mossa μ_i rimarrà proibita dopo la sua esecuzione. La tecnica di ricerca tabù fissa è ottenuta mantenendo costante il parametro T per tutta la durata della ricerca. In questo modo la scelta della soluzione S_{t+1} è effettuata nell'insieme $I_P(S_t) \subset I(S_t)$ delle soluzioni "permesse", che sono ottenibili cioè da S_t applicando una mossa che non è stata usata durante le ultime T iterazioni. In dettaglio, sia *last* $[\mu_i]$ l'iterazione t nella quale è stata usata per l'ultima volta la mossa μ_i (con *last* $[\mu_i]$ inizializzato a $-\infty$) e si indichino con μ_i^{-1} la mossa inversa di μ_i e con $c(S)$ il costo della soluzione S. Allora, si ha il seguente criterio, detto "della migliore mossa permessa":

$$\begin{aligned} I_P(S_t) = \{&S\text{: } S \text{ è ottenibile da } S_t \text{ con una mossa } \mu_i \in M \text{ tale che } \textit{last}\ [\mu_i^{-1}] <, \\ &t - T\ \} \\ S_{t+1} = S \in I_P(S_t) &\text{ tale che } c(S) \leq c(S') \text{ per ogni } S' \in I_P(S_t). \end{aligned}$$

Si noti che, poiché ci sono mosse proibite, non è detto che $c(S_{t+1}) \leq c(S_t)$, cioè, il costo della soluzione può peggiorare ed è così possibile sfuggire da un minimo locale, purché T sia scelto oculatamente. Chiaramente, se $T = 0$, allora non ci sono mai proibizioni e si riottiene la ricerca locale. In linea di principio, maggiore è T, maggiore è la diversificazione, cioè maggiore il numero di iterazioni prima che si rivisiti per la seconda volta la stessa soluzione. Però, T non può essere troppo grande, altrimenti tutte le mosse diventano presto proibite e la ricerca si blocca. In generale, il valore di T deve garantire l'esistenza di almeno due mosse permesse per ciascuna iterazione (altrimenti la ricerca non è neanche influenzata dal costo $c(S)$ delle soluzioni).

Supponiamo che una soluzione S sia individuata da una stringa di n bit e che la mossa μ_i consista nel complementare il bit i-esimo, $1 \leq i \leq n$. Si noti che ogni mossa è idempotente, cioè l'inversa di ogni mossa è uguale a se stessa: $\mu_i^{-1} = \mu_i$. Date due stringhe X ed Y, sia $H(X, Y)$ la loro *distanza di Hamming*, ovvero il numero di bit corrispondenti che sono diversi nelle due stringhe. In tal caso, $T \leq n-2$ garantisce che ci siano sempre almeno due mosse permesse. Se sono eseguite solo mosse permesse, allora è facile dimostrare che il minimo intervallo R di ripetizione è $2(T + 1)$. In altri termini, se $S_{t+R} = S_t$, allora $R \geq 2(T+1)$. La dimostrazione è immediata poiché, quando un bit è complementato, resta "congelato" per le prossime T iterazioni.

Esempio 1 (Proibizioni e diversificazione)**.** Sia la funzione da minimizzare $c(S)$ uguale al numero intero la cui rappresentazione binaria è data proprio da S (in tal caso, la soluzione ottima S^* è data dalla

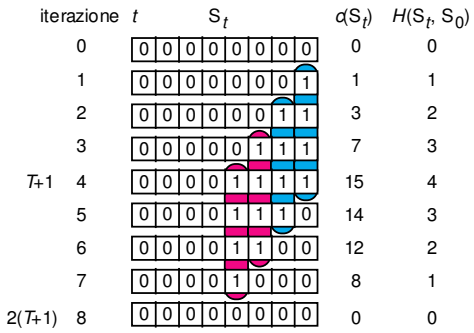

Figura 1: Correlazione tra proibizione T e diversificazione $H(S_t, S_0)$. Nell'esempio, $T = 3$.

stringa nulla). Come illustrato nella Fig. 1 per $n = 8$, si assuma di partire proprio da S^* e sia $T = 3$. All'iterazione 0 la mossa che dà il minimo valore della funzione c è quella che complementa il bit meno significativo. All'iterazione 1, il bit meno significativo è "proibito" (il periodo in cui un bit è "congelato", perché la rispettiva mossa è proibita, è mostrato in nero) e quindi la migliore mossa permessa consiste nel cambiare il penultimo bit meno significativo, e così via. Si noti che la distanza di Hamming è crescente nelle prime $T + 1$ iterazioni e che la massima distanza di Hamming è raggiunta all'iterazione $T + 1$. Successivamente, la distanza decresce ed all'iterazione $2(T + 1)$ si ripete la soluzione iniziale. Si noti comunque che le soluzioni visitate nelle prime $T + 1$ iterazioni (quando la distanza cresce) sono diverse da quelle visitate nelle successive $T + 1$ iterazioni (quando la distanza decresce). In questo modo, non si perde tempo a rivisitare soluzioni già visitate (a parte la soluzione iniziale). □

Vediamo come progettare una semplice euristica di "ricerca tabù fissa" per la BISEZIONE, dove il parametro T di proibizione è fornito come dato di ingresso alla procedura e rimane inalterato durante la computazione.

Proibizione fissa

Consideriamo una euristica di ricerca tabù fissa per la BISEZIONE dove una soluzione $S = (V_0, V_1)$ è individuata da una stringa di n bit e la mossa μ_i consiste nel complementare il bit i-esimo, $1 \leq i \leq n$, come nell'Esempio 1. In questo modo, V_0 è rappresentato dagli indici dei bit a 0, V_1 dagli indici dei bit ad 1, e la mossa μ_i può essere individuata specificando semplicemente l'indice i. Si noti che, per definizione, la soluzione S non è necessariamente ammissibile. Infatti, poiché una mossa consiste di fatto nello spostare un nodo da V_0 ad V_1 o viceversa, non è detto in generale che V_0 ed V_1 abbiano la stessa cardinalità. Solo se ciò avviene, cioè se $|V_0| = |V_1| = n/2$, allora la soluzione S è ammissibile.

Nel seguito, verrà utilizzato un parametro di proibizione frazionale T_f tale che $T = \lfloor nT_f \rfloor$, in modo da "normalizzare" la proibizione rispetto ad n. Inoltre, viene usata come contatore di iterazione una variabile globale t accedibile e modificabile da tutte le procedure. Analogamente, c_{min} è una variabile globale che indica il costo $c(S_{min})$ della migliore soluzione ammissibile S_{min} incontrata durante la ricerca. Si assume che tale variabile sia inizializzata dalla procedura *min-max-greedy* (). Anche *last* è un vettore globale, tale che $last[i]$ registra l'ultimo valore della variabile di iterazione t per il quale il nodo i è stato spostato da un insieme all'altro. Infine, sono usate due variabili n_0 ed n_1 tali che $n_0 = |V_0|$ e $n_1 = |V_1|$.

La seguente procedura bisectionFixed() riceve come parametri la proibizione frazionale T_f ed il numero di iterazioni da eseguire. Si assume che una soluzione ammissibile di partenza sia stata individuata

con *min-max-greedy*. La procedura bisectionFixed() sposta un nodo alla volta. L'indice k individua il sottoinsieme V_k nel quale aggiungere il nodo i tolto da V_{1-k}. Il nodo i è determinato dalla funzione bestMove() in accordo al criterio della "migliore mossa permessa", cioè in modo che $last[i] < t - T$ e il costo c sia il più piccolo possibile spostando i da V_{1-k} a V_k (ciò può essere fatto efficientemente utilizzando una struttura di dati "a cestini" (bucket). Se la soluzione $S = (V_0, V_1)$ è ammissibile ed il costo $c(S)$ è minore del costo c_{min} della migliore soluzione finora trovata, allora viene registrato il costo della nuova soluzione.

```
bisectionFixed(real T_f, integer iterations)
    T ← ⌊nT_f⌋
    for integer j ← 1 to iterations do
        k ← iif(n_0 > n/2, 1, 0)
        integer i ← bestMove(1 − k)
        V_k ← V_k ∪ {i}
        V_{1−k} ← V_{1−k} − {i}
        last[i] ← t
        t ← t + 1
        if n_0 = n_1 and c(S) < c_min then c_min ← c(S)
```

A causa della struttura stessa del problema della BISEZIONE, è bene non scegliere valori di T_f maggiori di $1/4$. Infatti una stringa binaria ed il suo complemento (con gli zeri e gli uno tra loro scambiati) indicano di fatto la stessa soluzione, con i due insiemi della partizione scambiati. Pertanto, un valore di $T_f > 1/2$ comporterebbe che, dopo essere partiti con una soluzione, si raggiungerebbe dopo $T + 1$ iterazioni una partizione più vicina al complemento che a quella di partenza. La scelta di $T_f < 1/4$ elimina queste "oscillazioni" tra una soluzione ed il suo complemento. Una scelta ragionevole del parametro *iterations* è invece $100n$. Prove sperimentali hanno mostrato come la scelta di T_f abbia un effetto cruciale sulla soluzione trovata: una scelta "azzeccata" porta a buone bisezioni, mentre una scelta "sbagliata" porta a bisezioni scadenti.

Proibizione randomizzata

Per evitare di dover provare e riprovare "manualmente" svariati valori di T_f eseguendo ripetutamente la procedura bisectionFixed() fino a selezionare il valore di T_f buono, si possono semplicemente effettuare delle scelte casuali! Si ottiene così la procedura probabilistica bisectionRandom().

```
bisectionRandom(integer iterations, integer individuale)
    t ← 0
    while t < iterations do
        min-max-greedy
        t_end ← t + individuale
        while t < t_end do
            T_f ← random(1, 25)/100
            bisectionFixed(T_f, n)
```

Il parametro *iterations* fornisce il numero complessivo di iterazioni, mentre *individuale* indica il numero di iterazioni di ogni singola "fase", che riparte dopo aver richiamato *min-max-greedy* (se *individuale* = *iterations* allora avviene una sola fase). Ogni n iterazioni viene scelto casualmente un nuovo T_f. In questo modo, si evita di scegliere un "cattivo" valore di T_f e mantenerlo per molte iterazioni. Al contrario, si continua a scegliere sempre valori diversi di T_f, che sono mantenuti per poche iterazioni. Prove sperimentali hanno mostrato che, scegliendo *individuale* $= 10n$ ed *iterations* $= 100n$, le soluzioni trovate da questa "ricerca tabù randomizzata" sono paragonabili a quelle trovate dalla "ricerca tabù fissa" nella quale si usi il miglior valore di T_f! In altri termini, molti valori casuali usati per poco tempo danno gli stessi risultati del miglior valore possibile usato a lungo.

Per trovare soluzioni ancora migliori del problema della BISEZIONE, si può impiegare una *ricerca tabù reattiva*, dove la messa a punto del parametro T_f è effettuata automaticamente dalla procedura stessa, sempre usando la randomizzazione, tenendo conto delle proprietà del grafo in ingresso e dello spazio delle soluzioni. I dettagli non sono qui riportati e possono essere reperiti nell'articolo citato nella bibliografia.