Algoritmi Branch-and-Bound: Strategie di Esplorazione e Rilassamenti[1]

Silvano Martello
Dipartimento di Informatica, Università di Torino

1 Branch-and-bound

Il metodo *branch-and-bound* è oggi lo strumento di più ampio uso per la definizione di algoritmi esatti per problemi di ottimizzazione combinatoria. Un problema viene detto di *ottimizzazione* quando consiste nella scelta, tra un numero finito o infinito di soluzioni ammissibili, di una soluzione migliore di tutte le altre secondo un criterio prefissato. Quando il numero di soluzioni ammissibili è finito, il problema viene detto di *ottimizzazione combinatoria*. In via del tutto teorica, questi problemi potrebbero ovviamente essere risolti da un algoritmo che enumerasse tutte le soluzioni ammissibili, scegliendo quella ottima; poichè però il numero di soluzioni ammissibili cresce di solito esponenzialmente nella dimensione delle istanze, tale approccio non è evidentemente praticabile per la soluzione effettiva di problemi di interesse pratico. Per una piccola parte dei problemi di ottimizzazione combinatoria sono noti algoritmi che consentono di risolvere ogni istanza del problema in un tempo limitato da una funzione polinomiale della dimensione dell'istanza stessa, mentre per la maggior parte di essi (appartenente alla classe dei problemi *NP-difficili*), non solo tali algoritmi non sono noti, ma anzi se ne ritiene estremamente improbabile l'esistenza. Le tecniche branch-and-bound consentono, per tali problemi, una enumerazione "intelligente" dell'insieme delle soluzioni ammissibili: nonostante il tempo di soluzione rimanga esponenziale nel caso peggiore, molti problemi possono essere "quasi sempre" risolti in tempi contenuti, anche nel caso di istanze di elevata dimensione.

Sia S l'insieme delle soluzioni ammissibili e

$$z : S \longrightarrow R^1$$

una funzione, detta *funzione obiettivo*, che associa a ciascun elemento $y \in S$ un valore $z(y)$. La coppia (z, S) definisce il problema di ottimizzazione, consistente nell'individuare l'elemento $y^* \in S$ tale che

$$z(y^*) \geq z(y) \quad \forall y \in S.$$

[1]Estratto da *Metodi di ottimizzazione per le decisioni*, a cura di G. Di Pillo, Masson, Milano, 1994.

Nel seguito si farà sempre riferimento a problemi di massimizzazione; i risultati si estendono banalmente alla minimizzazione (in cui si richiede $z(y^*) \leq z(y) \ \ \forall y \in S$).

Sia P^0, definito dalla coppia $(z, S(P^0))$, il problema da risolvere, e $Z(P^0)$ il valore della sua soluzione ottima, cioè

$$Z(P^0) = \max\{z(y) : y \in S(P^0)\}.$$

Il metodo branch-and-bound è basato sulla suddivisione di P^0 in un certo numero di sottoproblemi $P^1, P^2, \ldots, P^{n_0}$, la cui totalità rappresenti P^0. Ciò viene ottenuto suddividendo $S(P^0)$ in sottoinsiemi $S(P^1), S(P^2), \ldots, S(P^{n_0})$ tali che

$$\bigcup_{k=1}^{n_0} S(P^k) = S(P^0)$$

e definendo $Z(P^k) = \max\{z(y) : y \in S(P^k)\}$ $(k = 1, \ldots, n_0)$. Poichè qualunque soluzione ammissibile di P^0 è anche soluzione ammissibile di (almeno) uno tra $P^1, P^2, \ldots, P^{n_0}$, si avrà evidentemente

$$Z(P^0) = \max\{Z(P^1), Z(P^2), \ldots, Z(P^{n_0})\}.$$

Anzichè risolvere P^0 si risolveranno quindi i sottoproblemi $P^k (k = 1, \ldots, n_0)$, intendendo per risolvere:

1. determinare la soluzione ottima di P^k, oppure

2. dimostrare che P^k è impossibile, cioè che $S(P^k) = \emptyset$, oppure

3. dimostrare che la soluzione ottima di P^k non può essere migliore della miglior soluzione ammissibile di P^0 eventualmente ottenuta in precedenza.

Il vantaggio del procedimento consiste nell'ottenere problemi "più facili" in quanto relativi ad insiemi con un minor numero di soluzioni ammissibili (secondo il noto principio *"divide et impera"*). È pertanto generalmente preferibile che la suddivisione di $S(P^0)$ costituisca una partizione, cioè che si abbia anche

$$S(P^i) \cap S(P^j) = \emptyset \ \ \forall \ P^i, P^j : i \neq j.$$

In generale sarà comunque impossibile risolvere facilmente uno o più dei sottoproblemi ottenuti. La suddivisione verrà allora ripetuta per ogni sottoproblema che non è stato risolto. Il procedimento viene di solito rappresentato dinamicamente mediante un *albero decisionale* quale quello di Figura 1, donde l'adozione del termine ramificazione (*branching*) per il procedimento stesso, e della terminologia tradizionale relativa agli alberi: nodo radice (P^0), nodi (o problemi) padri o figli, generazione di nodi, ecc.

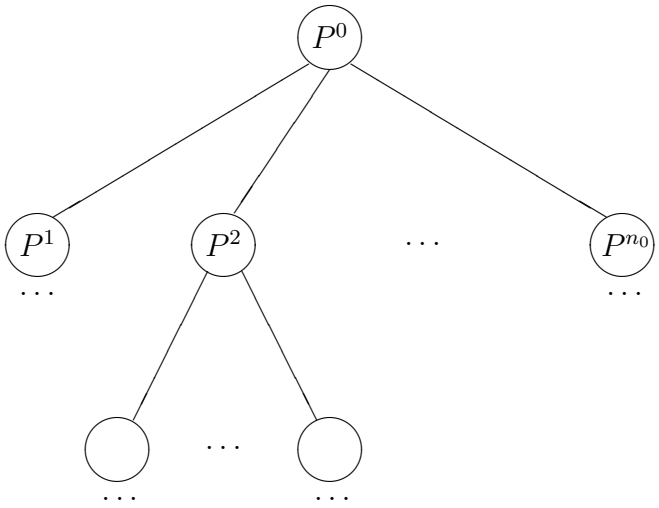

Figura 1

Ad un qualunque livello di sviluppo l'insieme di sottoproblemi da risolvere è quello corrispondente ai nodi estremi (foglie) dell'albero decisionale. La scelta del sottoproblema sul quale operare la successiva ramificazione viene effettuata seguendo una prefissata strategia: le diverse strategie utilizzate sono discusse nella Sezione 2.

Qualunque sia la strategia di ramificazione prescelta, è essenziale per l'efficienza dell'algoritmo che una elevata percentuale dei sottoproblemi generati possa essere risolta senza ricorrere ad ulteriori ramificazioni. A tal fine riveste fondamentale importanza il criterio di soluzione 3: esso viene realizzato calcolando, per ogni sottoproblema P^k affrontato, un limite superiore (*upper bound*) sul valore della soluzione del sottoproblema stesso, ossia un valore $U(P^k)$ tale che

$$U(P^k) \geq Z(P^k).$$

Detto Z il valore della miglior soluzione ammissibile nota, il criterio 3 può dunque essere riscritto come

3. verificare che $U(P^k) \leq Z$.

Un upper bound viene usualmente ottenuto calcolando la soluzione ottima di un *rilassamento* del problema. Si dice rilassamento di un problema P^k, definito da $(z, S(P^k))$, qualunque problema $R(P^k)$ definito da $(z_r, S_r(P^k))$ soddisfacenti

(A) $S_r(P^k) \supseteq S(P^k)$,

(B) $z_r(y) \geq z(y) \quad \forall y \in S(P^k)$.

Risulta evidente da (A), (B) che se $R(P^k)$ non ammette soluzione altrettanto vale per

P^k, e che altrimenti il valore della soluzione ottima di $R(P^k)$ non può essere inferiore a quello della soluzione ottima di P^k. Si avrà dunque l'upper bound

$$U(P^k) = Z(R(P^k)) = \max\{z_r(y) : y \in S_r(P^k)\}.$$

La scelta del rilassamento da adottare dipende ovviamente dallo specifico problema affrontato e viene effettuata sulla base di due criteri, spesso contrastanti: (a) il valore di $Z(R(P^k))$ deve essere il più possibile prossimo a quello di $Z(P^k)$, ossia abbastanza basso da garantire che la condizione 3 si verifichi con adeguata frequenza; (b) $R(P^k)$ deve essere sufficientemente "facile" da risolvere (ed in ogni caso, ovviamente, più facile di P^k). Nella Sezione 3 vengono descritte le tecniche di rilassamento di uso più frequente per problemi di programmazione lineare intera.

2 Strategie di esplorazione

Le strategie di esplorazione attualmente più in uso possono essere ricondotte a due tipologie principali: *depth-first* e *highest-first*. Si rammenti che Z denota il valore della miglior soluzione ammissibile nota; inizialmente a Z viene dato il valore $-\infty$ oppure quello di una soluzione ammissibile determinata mediante un algoritmo approssimato.

2.1 Depth-first

Si inizia calcolando l'upper bound del nodo P^0, generando il suo primo figlio P^1 (lasciando in sospeso la generazione dei rimanenti figli $P^2, \ldots, P^{n_0}$) e calcolando l'upper bound di P^1. Si genera quindi il primo figlio di P^1, e così via, producendo una sequenza di passi avanti (*forward step*) in cui viene sempre scelto per la generazione l'ultimo nodo generato, finchè si ottiene un problema che può essere facilmente risolto (nel qual caso si provvede all'eventuale aggiornamento di Z), oppure l'upper bound del nodo generato è non maggiore di Z, oppure il nodo attuale non possiede figli ammissibili. A questo punto si effettua un passo indietro (*backtracking*): si risale al penultimo nodo generato e, se il valore del relativo upper bound (già calcolato al momento della generazione) è maggiore di Z, si genera il suo figlio successivo, quindi il primo figlio di questi, e così via; altrimenti (o se tutti i figli del penultimo nodo sono già stati generati), si risale al nodo precedente. In altri termini, i problemi generati vengono posti in una pila: si genera sempre un figlio del primo problema della pila e lo si pone al primo posto; quando un problema viene risolto lo si rimuove dalla pila. L'esecuzione termina quando la pila è vuota, cioè quando si dovrebbe effettuare un backtracking dal nodo radice.

Con questa strategia il numero di nodi attivi (nodi generati e dai quali occorre ramificare ulteriormente) si mantiene relativamente basso nel corso dell'esecuzione, non potendo superare il numero di livelli dell'albero. Inoltre ogni nodo è padre o figlio del nodo precedentemente trattato, per cui la definizione del corrispondente sottoproblema risulta agevole e non richiede di solito l'introduzione di strutture-dati complesse. Conseguentemente la stesura degli algoritmi e dei codici di calcolo corrispondenti è più semplice di quanto avvenga per le altre strategie di esplorazione. Un ulteriore vantaggio della strategia depth-first è quello di produrre rapidamente soluzioni ammissibili, di valore via via crescente, e pertanto di produrre, anche nel caso di arresto anticipato dell'esecuzione (ad esempio per raggiunto limite di tempo), buone soluzioni approssimate.

2.2 Highest-first

Si inizia calcolando l'upper bound del nodo P^0 ed inserendo il nodo stesso in un insieme di nodi attivi. Ad ogni iterazione si rimuove da tale insieme il nodo dotato del più alto upper bound, si generano tutti i suoi figli e si calcolano i corrispondenti upper bound: i nodi non risolti che hanno upper bound maggiore di Z vengono inseriti nell'insieme dei nodi attivi e, nel caso ci siano nodi risolti, si provvede all'eventuale aggiornamento di Z. Si termina quando l'insieme dei nodi attivi è vuoto o contiene solo nodi con upper bound non maggiore di Z.

Il principale vantaggio di questa strategia è dato dalla scelta di esplorare, ad ogni iterazione, il nodo che, fra tutti quelli generati sino a quel momento, ha la massima probabilità di produrre una soluzione di valore elevato; di conseguenza il numero di nodi complessivamente esplorati dall'algoritmo è normalmente inferiore a quello richiesto dalla strategia depth-first. D'altra parte la necessità di mantenere memorizzati un elevato numero di sottoproblemi e l'assenza di qualunque relazione tra il sottoproblema corrente e quello precedentemente considerato comportano l'utilizzazione di strutture-dati più complesse ed un aumento dell'occupazione di memoria e del tempo di calcolo necessario per l'esplorazione di ciascun nodo. Inoltre l'algoritmo difficilmente produce soluzioni ammissibili prima che sia stato espolorato un elevato numero di nodi.

Nel caso di problemi di minimizzazione, la strategia, basata ovviamente sulla scelta del limite inferiore (*lower bound*) di valore minimo, viene detta *lowest-first*. Per entrambi i casi si usa anche (meno frequentemente) il termine *best-bound-first*.

In Figura 2 è rappresentato un esempio di albero decisionale: i valori in grassetto accanto ai nodi sono quelli dei corrispondenti upper bound, mentre un valore di Z al di sotto di un nodo indica che esso produce una soluzione ammissibile di tale valore.

All'interno dei nodi sono date le sequenze di esplorazione della strategia depth-first (A, B, C, ...) e della strategia highest-first (1, 2, 3, ...). Con la prima strategia vengono calcolati gli upper bound dei nodi effettivamente esplorati, mentre con la seconda vengono calcolati gli upper bound di tutti i nodi ad eccezione di C, D, E, F.

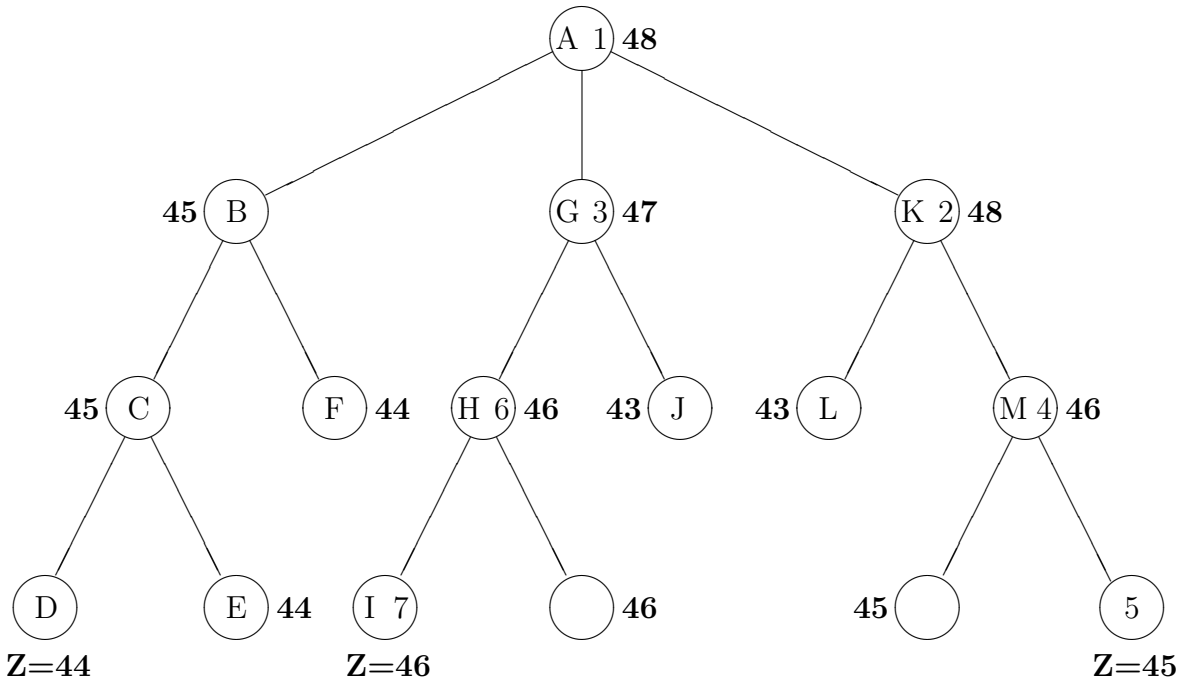

Figura 2

2.3 Altre strategie

Una strategia *ibrida* delle due sopra descritte si ottiene dalla strategia depth-first generando, ad ogni passo avanti, tutti i figli del nodo prescelto, calcolando i valori dei corrispondenti upper bound e proseguendo l'esplorazione dal figlio corrispondente al massimo upper bound. Dopo un backtracking, l'esplorazione riprende dal figlio con massimo upper bound tra quelli non ancora esplorati. Si ottiene in questo modo una parte del vantaggio della strategia highest-first (esplorazione più "intelligente" dell'albero decisionale), a spese di un aumento del numero di upper bound calcolati, mantenendo strutture-dati relativamente semplici (in quanto ogni nodo è padre o figlio del nodo precedentemente trattato) ed ottenendo ancora soluzioni ammissibili dopo un numero non elevato di nodi esplorati. Nell'esempio di Figura 2 la sequenza di esplorazione della strategia ibrida sarebbe 1, 2, 4, 5, 3, 6, 7 e verrebbero calcolati gli upper bound di tutti i nodi ad eccezione di C, D, E, F.

Menzioniamo infine la strategia *breadth-first*, oggi raramente utilizzata. Si inizia calcolando l'upper bound del nodo P^0, generando tutti i suoi figli e calcolando i corrispondenti upper bound. Vengono quindi generati tutti i figli di ogni figlio di P^0 che non può essere risolto e che ha upper bound maggiore di Z, e così via, esplorando sempre tutti i nodi di un livello prima di passare al livello successivo. La complessità dell'algoritmo risultante, l'elevato numero di nodi esplorati e la grande occupazione di memoria spiegano lo scarso successo (salvo applicazioni particolari, quali ad esempio l'enumerazione di tutte le soluzioni ammissibili di valore non inferiore ad una soglia prefissata) di questa strategia. Segnaliamo infine che alcuni autori utilizzano il termine *breadth-first* come sinonimo di *best-bound-first*.

3 Rilassamenti

Come si è detto nella Sezione 1, gli upper bound necessari per il buon comportamento computazionale di un algoritmo branch-and-bound vengono dati dalla soluzione di un rilassamento del problema dato, ottenuto ingrandendo l'insieme delle soluzioni ammissibili e/o sostituendo la funzione obiettivo con una funzione che assuma valore non minore di essa per ogni soluzione ammissibile del problema stesso. Nella presente sezione vengono esaminate le tecniche di rilassamento di uso più frequente per problemi di ottimizzazione combinatoria; nella sezione successiva tali tecniche vengono confrontate tra di loro, al fine di stabilire eventuali dominanze dell'una sull'altra, dal punto di vista della qualità degli upper bound prodotti.

La formulazione matematica più naturale di un problema di ottimizzazione combinatoria è un modello di programmazione a variabili intere o booleane. Molto spesso la funzione obiettivo è lineare e l'insieme delle soluzioni ammissibili è esprimibile mediante funzioni lineari. Utilizzando per semplicità (e senza perdita di generalità) variabili booleane, considereremo pertanto il problema P definito da:

$$Z(P) = \max \sum_{j=1}^{n} c_j x_j \tag{1}$$

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad (i = 1, \ldots, m), \tag{2}$$

$$\sum_{j=1}^{n} d_{kj} x_j = e_k \qquad (k = 1, \ldots, l), \tag{3}$$

$$x_j \in \{0, 1\} \qquad (j = 1, \ldots, n). \tag{4}$$

Gli elementi ammissibili $y \in S$ sono quindi rappresentati da un insieme di punti nello spazio $\{0, 1\}^n$, ossia dai vettori booleani $x = (x_1, \ldots, x_n)$ che soddisfano (2), (3). Il

modello potrebbe essere semplificato utilizzando soli vincoli di tipo (2) oppure di tipo (3), in quanto i vincoli di disuguaglianza possono essere facilmente trasformati in equivalenti vincoli di uguaglianza e viceversa; manterremo tuttavia entrambi i tipi al fine di meglio evidenziare alcune caratteristiche dei rilassamenti che saranno trattati.

3.1 Rilassamento per eliminazione

Il modo più semplice per ottenere un rilassamento consiste nell'eliminare uno o più vincoli del modello. Ad esempio, eliminando da P i vincoli di uguaglianza si ottiene il problema $E(P)$:

$$\begin{aligned} Z(E(P)) = \quad \max \quad & \sum_{j=1}^{n} c_j x_j \\ & \sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad (i = 1, \ldots, m), \\ & x_j \in \{0, 1\} \qquad (j = 1, \ldots, n), \end{aligned}$$

noto come problema *knapsack multidimensionale 0-1*, che, specie nel caso di bassi valori di m, risulta di più semplice trattazione in quanto meglio strutturato. Eliminando anche tutti i vincoli di disuguaglianza tranne uno (ad esempio l'h-esimo) si ottiene il problema *knapsack singolo 0-1*, che può spesso essere risolto agevolmente:

$$\begin{aligned} Z(E(P)) = \quad \max \quad & \sum_{j=1}^{n} c_j x_j \\ & \sum_{j=1}^{n} a_{hj} x_j \leq b_h, \\ & x_j \in \{0, 1\} \qquad (j = 1, \ldots, n). \end{aligned}$$

Si osservi che se la soluzione ottima x^* di $E(P)$ è ammissibile per P (cioè rispetta i vincoli eliminati), allora tale soluzione è ottima per P.

3.2 Rilassamento continuo

Un altro semplice rilassamento di P si ottiene dalla rimozione dei vincoli di interezza, che produce il problema $C(P)$:

$$Z(C(P)) = \quad \max \quad \sum_{j=1}^{n} c_j x_j$$

$$\begin{aligned}
&\sum_{j=1}^{n} a_{ij}x_j \leq b_i \qquad (i = 1, \ldots, m),\\
&\sum_{j=1}^{n} d_{kj}x_j = e_k \qquad (k = 1, \ldots, l),\\
&0 \leq x_j \leq 1 \qquad (j = 1, \ldots, n),
\end{aligned}$$

ossia un problema di *programmazione lineare*, risolubile in tempo polinomiale. Se i coefficienti c_j della funzione obiettivo sono interi (come si verifica frequentemente), l'upper bound $Z(C(P))$ può evidentemenete essere migliorato in $\lfloor Z(C(P)) \rfloor$.

Nella Sezione 1 è stato osservato, per un generico problema P, che se un rilassamento $R(P)$ non ammette soluzione altrettanto vale per P, mentre se $R(P)$ è illimitato (cioè $Z(R(P)) = +\infty$) nulla si può affermare in generale su P. Nel caso del rilassamento continuo è però facile verificare che se $C(P)$ è illimitato allora P è illimitato o impossibile.

Se la soluzione ottima x^* di $C(P)$ è ammissibile per P (se cioè si ha x_j intero $\forall j$), allora tale soluzione è ottima per P.

3.3 Rilassamento surrogato

Questo rilassamento viene frequentemente applicato a vincoli di disuguaglianza ed ha l'effetto di sostituire un insieme di vincoli con un unico vincolo di uguale struttura. Si considerino gli m vincoli (2) di P e si introduca un vettore π di m componenti $\pi_i \geq 0$ (detti *moltiplicatori surrogati*). Si moltiplichino entrambi i membri di ciascun vincolo per il corrispondente moltiplicatore e si sommino tra loro tutti i membri sinistri e tutti i membri destri, ottenendo

$$\sum_{i=1}^{m} \pi_i \sum_{j=1}^{n} a_{ij}x_j \leq \sum_{i=1}^{m} \pi_i b_i. \tag{5}$$

Si aggiunga (5) a P e si osservi che il problema resta inalterato, in quanto ogni x soddisfacente (2) soddisfa anche (5). Eliminando ora i vincoli (2), si ottiene il rilassamento surrogato $S(P, \pi)$:

$$\begin{aligned}
Z(S(P,\pi)) = \quad \max \quad & \sum_{j=1}^{n} c_j x_j\\
& \sum_{j=1}^{n} \hat{a}_j x_j \leq \hat{b},\\
& \sum_{j=1}^{n} d_{kj}x_j = e_k \qquad (k = 1, \ldots, l),\\
& x_j \in \{0, 1\} \qquad (j = 1, \ldots, n),
\end{aligned}$$

dove

$$\begin{aligned} \hat{a}_j &= \sum_{i=1}^{m} \pi_i a_{ij} \qquad (j = 1, \ldots, n), \\ \hat{b} &= \sum_{i=1}^{m} \pi_i b_i. \end{aligned}$$

Se la soluzione ottima x^* di $S(P, \pi)$ è ammissibile per P (cioè rispetta i vincoli (2)), allora tale soluzione è ottima per P.

Il rilassamento fornisce un upper bound su $Z(P)$ qualunque sia il vettore $\pi \geq 0$ prescelto (utilizzando ad esempio $\pi_i = 0 \; \forall i$, si ottiene un rilassamento per eliminazione). Poichè un upper bound è tanto più utile quanto più il suo valore è basso, si pone quindi il *problema surrogato duale* di determinare il vettore π^* tale che

$$Z(S(P, \pi^*)) = \min_{\pi \geq 0} \{Z(S(P, \pi))\},$$

o, quando ciò non sia possibile, un vettore non negativo π che produca valori sufficientemente bassi di $Z(S(P, \pi))$.

3.4 Rilassamento lagrangiano di vincoli di disuguaglianza

Si rammenti che il rilassamento di un problema si ottiene in generale modificando la funzione obiettivo e/o l'insieme delle soluzioni ammissibili. I rilassamenti visti sinora aggiungono soluzioni ammissibili mantenendo inalterata la funzione obiettivo. I rilassamenti lagrangiani sono invece basati su una eliminazione di vincoli accompagnata da una modifica della funzione obiettivo tale da far sì che la soluzione del problema rilassato continui a dipendere in qualche misura dai vincoli eliminati e tenda a non violarli eccessivamente.

Si considerino gli m vincoli (2) di P e si introduca un vettore λ di m componenti $\lambda_i \geq 0$ (*moltiplicatori lagrangiani*). Se si mantengono inalterati i vincoli di P e si modifica la funzione obiettivo in

$$\max \sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{m} \lambda_i (b_i - \sum_{j=1}^{n} a_{ij} x_j), \tag{6}$$

si ottiene un rilassamento valido del problema. Infatti (si rammenti la condizione (B) della Sezione 1), per qualunque soluzione x soddisfacente (2) si ha $b_i - \sum_{j=1}^{n} a_{ij} x_j \geq 0 \; \forall i$, per cui il valore della funzione obiettivo modificata non è inferiore a quello della funzione originaria. Eliminando da questo rilassamento i vincoli (2) (e manipolando

algebricamente la funzione obiettivo in maniera tale da conservare la struttura della funzione originaria), si ottiene allora il rilassamento lagrangiano $L(P, \lambda)$:

$$\begin{aligned} Z(L(P,\lambda)) = \sum_{i=1}^{m} \lambda_i b_i + \quad \max \quad & \sum_{j=1}^{n} \hat{c}_j x_j \\ & \sum_{j=1}^{n} d_{kj} x_j = e_k \qquad (k = 1, \ldots, l), \\ & x_j \in \{0, 1\} \qquad (j = 1, \ldots, n), \end{aligned}$$

dove

$$\hat{c}_j \quad = \quad c_j - \sum_{i=1}^{m} \lambda_i a_{ij} \qquad (j = 1, \ldots, n).$$

Osservando la funzione obiettivo nella forma (6) si nota chiaramente come ogni violazione di un vincolo (2) provochi una diminuzione del valore della funzione stessa, per cui la soluzione ottima di $L(P, \lambda)$ viene in un certo senso forzata a non trascurare troppo i vincoli eliminati.

Se la soluzione ottima x^* di $L(P, \lambda)$ è ammissibile per P (cioè rispetta i vincoli (2)), non è detto che tale soluzione sia ottima per P (contrariamente a quanto avviene per i rilassamenti in cui la funzione obiettivo resta invariata), in quanto la funzione obiettivo del problema rilassato è, in generale, diversa da quella originaria. Se però x^* soddisfa anche

$$\sum_{i=1}^{m} \lambda_i (b_i - \sum_{j=1}^{n} a_{ij} x_j^*) = 0,$$

allora essa è ottima per P, in quanto in tale punto le due funzioni coincidono. In altri termini, la soluzione ottima di $L(P, \lambda)$ è ottima per P se rispetta tutti i vincoli ed ha moltiplicatore nullo per ogni vincolo lasco.

Anche per questo rilassamento si ha interesse a determinare il miglior vettore di moltiplicatori, cioè (*problema lagrangiano duale*) il vettore λ^* tale che

$$Z(L(P, \lambda^*)) = \min_{\lambda \geq 0} \{Z(L(P, \lambda))\}.$$

Quando tale problema non può essere risolto per via analitica, le proprietà sopra evidenziate consentono di realizzare facilmente tecniche iterative per la determinazione di buoni moltiplicatori lagrangiani. Sia infatti $\lambda^q = (\lambda_1^q, \lambda_2^q, \ldots, \lambda_m^q)$ il vettore corrente e sia x^* la soluzione ottima di $L(P, \lambda^q)$: se x^* non è ottima per P, sarà conveniente utilizzare, nella successiva iterazione, il vettore λ^{q+1} definito da

$$\lambda_i^{q+1} = \max(0, \lambda_i^q - \delta^q (b_i - \sum_{j=1}^{n} a_{ij} x_j^*)),$$

dove δ^q è un opportuno passo di correzione positivo; la modifica infatti aumenta il valore dei moltiplicatori corrispondenti a vincoli violati, diminuisce il valore di quelli corrispondenti a vincoli laschi e lascia invariati i rimanenti. Il metodo qui accennato (che è una versione semplificata della nota *tecnica del subgradiente*), pur producendo in generali buoni risultati, non garantisce la convergenza monotona dei bound ottenuti, ossia non si ha necessariamente $Z(L(P, \lambda^{q+1})) \leq Z(L(P, \lambda^q))$.

3.5 Rilassamento lagrangiano di vincoli di uguaglianza

Si considerino gli l vincoli (3) di P e si definisca, analogamente a quanto fatto nella sezione precedente, la funzione obiettivo modificata:

$$\max \sum_{j=1}^{n} c_j x_j + \sum_{k=1}^{l} \lambda_k (e_k - \sum_{j=1}^{n} d_{kj} x_j); \tag{7}$$

si osservi che, per qualunque soluzione x soddisfacente (3), si ha $e_k - \sum_{j=1}^{n} d_{kj} x_j = 0$, ossia la funzione coincide con quella originaria. Non è pertanto necessario in questo caso imporre la non negatività dei moltiplicatori λ_k. Eliminando i vincoli (3) otterremo il rilassamento lagrangiano:

$$\begin{aligned} Z(L(P,\lambda)) = \sum_{k=1}^{l} \lambda_k e_k + \quad \max \quad & \sum_{j=1}^{n} \hat{c}_j x_j \\ & \sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad (i = 1, \ldots, m), \\ & x_j \in \{0, 1\} \qquad (j = 1, \ldots, n), \end{aligned}$$

dove

$$\hat{c}_j \quad = \quad c_j - \sum_{k=1}^{l} \lambda_k d_{kj} \qquad (j = 1, \ldots, n).$$

Se la soluzione ottima x^* di $L(P, \lambda)$ è ammissibile per P (cioè rispetta i vincoli (3)), essa è in questo caso ottima per P, in quanto il contributo lagrangiano alla funzione obiettivo è nullo. Le rimanenti considerazioni della sezione precedente si estendono facilmente al caso qui considerato.

3.6 Rilassamento per decomposizione

In alcuni problemi di ottimizzazione combinatoria è possibile partizionare l'insieme dei vincoli in due sottoinsiemi tali che i rilassamenti per eliminazione di uno qualunque dei

due sottoinsiemi risultano di facile soluzione. In tali situazioni, determinando separatamente i valori delle soluzioni dei due problemi rilassati, si ottiene, dal minimo dei due, un upper bound valido per il problema. Il rilassamento per decomposizione consente invece di ottenere un unico upper bound (migliore), mediante una somma pesata delle due funzioni obiettivo opportunamente modificate.

Si supponga che il problema P sia di facile soluzione se i vincoli (2), oppure i vincoli (3), vengono eliminati. Introducendo n nuove variabili booleane y_j e prefissando due parametri α, β tali che $\alpha + \beta = 1$, il problema può essere riscritto nella forma:

$$
\begin{aligned}
Z(P') = \quad \max \quad & \alpha \sum_{j=1}^{n} c_j x_j + \beta \sum_{j=1}^{n} c_j y_j && (8)\\
& \sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad (i = 1, \ldots, m), && (9)\\
& \sum_{j=1}^{n} d_{kj} y_j = e_k \qquad (k = 1, \ldots, l), && (10)\\
& y_j = x_j \qquad (j = 1, \ldots, n), && (11)\\
& x_j \in \{0, 1\} \qquad (j = 1, \ldots, n), && (12)\\
& y_j \in \{0, 1\} \qquad (j = 1, \ldots, n). && (13)
\end{aligned}
$$

Risulta evidente dai vincoli aggiuntivi (11) che $Z(P') = Z(P)$ (utilizzando valori α, β qualunque, si avrebbe ancora un problema equivalente in cui $Z(P') = (\alpha + \beta)Z(P)$). Rilassando in maniera lagrangiana i vincoli (11) mediante moltiplicatori λ_j $(j = 1, \ldots, n)$ di segno qualunque, e manipolando come di consueto la funzione obiettivo, si ottiene quindi il rilassamento per decomposizione $D(P, \lambda)$:

$$
\begin{aligned}
Z(D(P, \lambda)) = \quad \max \quad & \sum_{j=1}^{n} (\alpha c_j + \lambda_j) x_j + \sum_{j=1}^{n} (\beta c_j - \lambda_j) y_j\\
& \sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad (i = 1, \ldots, m),\\
& \sum_{j=1}^{n} d_{kj} y_j = e_k \qquad (k = 1, \ldots, l),\\
& x_j \in \{0, 1\} \qquad (j = 1, \ldots, n),\\
& y_j \in \{0, 1\} \qquad (j = 1, \ldots, n),
\end{aligned}
$$

risolubile, grazie all'eliminazione dei vincoli (11), mediante ottimizzazione separata dei due termini della funzione obiettivo. In altre parole il problema si decompone in due problemi rilassati indipendenti:

a) un problema $DX(P,\lambda)$ nelle sole variabili x_j:

$$\begin{aligned} Z(DX(P,\lambda)) = \quad \max \quad & \sum_{j=1}^{n} \overline{c}_j x_j \\ & \sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad (i = 1, \ldots, m), \\ & x_j \in \{0,1\} \qquad (j = 1, \ldots, n), \end{aligned}$$

con $\overline{c}_j = \alpha c_j + \lambda_j$ $(j = 1, \ldots, n)$;

b) un problema $DY(P,\lambda)$ nelle sole variabili y_j:

$$\begin{aligned} Z(DY(P,\lambda)) = \quad \max \quad & \sum_{j=1}^{n} \tilde{c}_j y_j \\ & \sum_{j=1}^{n} d_{kj} y_j = e_k \qquad (k = 1, \ldots, l), \\ & y_j \in \{0,1\} \qquad (j = 1, \ldots, n), \end{aligned}$$

con $\tilde{c}_j = \beta c_j - \lambda_j$ $(j = 1, \ldots, n)$.

Le soluzioni di tali problemi produrranno l'upper bound su $Z(P)$:

$$Z(D(P,\lambda)) = Z(DX(P,\lambda)) + Z(DY(P,\lambda)).$$

Poichè $D(P,\lambda)$ è un rilassamento lagrangiano di vincoli di uguaglianza, se le soluzioni ottime x^*, y^* dei due problemi sono ammissibili per P' (cioè rispettano i vincoli (11)), allora x^* è ottima per P. Anche in questo caso si pone naturalmente il problema lagrangiano duale di determinare il vettore λ^* tale che

$$Z(D(P,\lambda^*)) = \min_{\lambda}\{Z(D(P,\lambda))\}.$$

4 Relazioni tra rilassamenti

Per la maggior parte dei rilassamenti del problema P descritti nella sezione precedente è possibile stabilire relazioni di dominanza, dal punto di vista della qualità degli upper bound prodotti, qualora si faccia riferimento al rilassamento dello stesso insieme di vincoli. Dato un rilassamento R ($R \in \{E, S, L\}$), la notazione $R_{(\mathrm{v})}$ indicherà pertanto che si considera il rilassamento dei vincoli (v) di P, mentre il rilassamento per decomposizione dei vincoli (v), (w) sarà indicato con $D_{(\mathrm{v}),(\mathrm{w})}$. Per i rilassamenti che prevedono un vettore μ di moltiplicatori, indicheremo inoltre con μ^* il vettore ottimo (ossia la soluzione del

problema duale di $R_{(\mathrm{v})[,(\mathrm{w})]}(P,\mu)$), e chiameremo *rilassamento duale* quello relativo a tale vettore.

Come osservato in precedenza, il rilassamento per eliminazione di un insieme qualunque (v) di vincoli è un caso particolare di rilassamento surrogato degli stessi vincoli, ossia $Z(E_{(\mathrm{v})}(P)) = Z(S_{(\mathrm{v})}(P,0))$; pertanto

Relazione 1 *Il rilassamento surrogato duale domina il rilassamento per eliminazione.*

Analogamente $Z(E_{(\mathrm{v})}(P)) = Z(L_{(\mathrm{v})}(P,0))$, da cui

Relazione 2 *Il rilassamento lagrangiano duale domina il rilassamento per eliminazione.*

Relazioni più complesse vengono analizzate nelle prossime sezioni.

4.1 Surrogato-Lagrangiano

Confrontando il rilassamento surrogato dei vincoli (2) di P,

$$
\begin{aligned}
Z(S_{(2)}(P,\mu)) = \quad \max \quad & \sum_{j=1}^{n} c_j x_j && (14)\\
& \sum_{i=1}^{m} \mu_i \sum_{j=1}^{n} a_{ij} x_j \leq \sum_{i=1}^{m} \mu_i b_i, && (15)\\
& \sum_{j=1}^{n} d_{kj} x_j = e_k \qquad (k = 1,\ldots,l), && (16)\\
& x_j \in \{0,1\} \qquad (j = 1,\ldots,n) && (17)
\end{aligned}
$$

ed il rilassamento lagrangiano degli stessi vincoli,

$$
\begin{aligned}
Z(L_{(2)}(P,\mu)) = \quad \max \quad & \sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{m} \mu_i b_i - \sum_{i=1}^{m} \mu_i \sum_{j=1}^{n} a_{ij} x_j\\
& \sum_{j=1}^{n} d_{kj} x_j = e_k \qquad (k = 1,\ldots,l),\\
& x_j \in \{0,1\} \qquad (j = 1,\ldots,n),
\end{aligned}
$$

risulta evidente che $L_{(2)}(P,\mu)$ può essere visto come il rilassamento lagrangiano del vincolo (15) di $S_{(2)}(P,\mu)$ con moltiplicatore unitario. Si avrà pertanto

$$Z(L_{(2)}(P,\mu)) \geq Z(S_{(2)}(P,\mu)) \ \forall \mu > 0,$$

ed analogamente,

$$Z(L_{(3)}(P,\mu)) \geq Z(S_{(3)}(P,\mu)) \ \forall \mu.$$

Poichè tali relazioni valgono ovviamente anche per il vettore dei moltiplicatori lagrangiani ottimi, si potrà dunque concludere che

Relazione 3 *Il rilassamento surrogato duale domina il rilassamento lagrangiano duale.*

Questa dominanza non implica però che il rilassamento surrogato sia da preferire al rilassamento lagrangiano. Si osservi infatti come quest'ultimo produca un problema meglio strutturato (in quanto contenente un vincolo in meno) e quindi, spesso, più facilmente risolubile. Inoltre il problema lagrangiano duale possiede proprietà teoriche che agevolano la ricerca di un buon vettore di moltiplicatori.

4.2 Lagrangiano-Continuo

La teoria della dualità nella programmazione lineare consente di provare che, dato il rilassamento lagrangiano $L(P,\lambda)$ di un qualunque sottoinsieme di vincoli di P, vale

$$Z(L(P,\lambda^*)) \leq Z(C(P)) \tag{18}$$

(si omette la dimostrazione), ossia che

Relazione 4 *Il rilassamento lagrangiano duale domina il rilassamento continuo.*

Dato invece un vettore λ qualunque, non esiste in generale nessuna relazione tra $Z(L(P,\lambda))$ e $Z(C(P))$. Esiste tuttavia un caso particolare, di notevole interesse, nel quale una relazione può essere stabilita: si introduca la seguente

Definizione 1 *Un modello di programmazione lineare intera Q possiede la* proprietà di interezza *se, per qualunque istanza, il suo rilassamento continuo ha soluzione ottima intera.*

Se il rilassamento lagrangiano di P possiede la proprietà di interezza si avrà pertanto

$$Z(L(P,\lambda)) = Z(C(L(P,\lambda))) \ \forall \lambda,$$

ossia, osservando che $C(L(P,\lambda))$ è un rilassamento di $C(P)$ e che pertanto

$$Z(C(L(P,\lambda))) \geq Z(C(P)) \ \forall \lambda,$$

si avrà

$$Z(L(P,\lambda)) \geq Z(C(P)) \ \forall\lambda; \tag{19}$$

da (18), (19) consegue dunque

$$Z(L(P,\lambda^*)) = Z(C(P)),$$

cioè

Relazione 5 *Se il rilassamento lagrangiano possiede la proprietà di interezza, il valore dato dal rilassamento lagrangiano duale coincide con quello dato dal rilassamento continuo.*

4.3 Surrogato-Continuo

Dalle relazioni 3 e 4 consegue ovviamente che

Relazione 6 *Il rilassamento surrogato duale domina il rilassamento continuo.*

Analogamente a quanto visto in precedenza, se il rilassamento surrogato di P possiede la proprietà di interezza si avrà $Z(S(P,\pi)) = Z(C(S(P,\pi))) \geq Z(C(P)) \ \forall\pi$ e dunque $Z(S(P,\pi^*)) = Z(C(P))$, cioè

Relazione 7 *Se il rilassamento surrogato possiede la proprietà di interezza, il valore dato dal rilassamento surrogato duale coincide con quello dato dal rilassamento continuo.*

4.4 Decomposizione-Lagrangiano

È possibile provare che

$$Z(D_{(\mathrm{v}),(\mathrm{w})}(P,\lambda^*)) \leq \min(Z(L_{(\mathrm{v})}(P,\mu^*)), Z(L_{(\mathrm{w})}(P,\nu^*)))$$

(si omette la dimostrazione), ossia che

Relazione 8 *Il rilassamento per decomposizione duale domina il rilassamento lagrangiano duale.*

Non esistono invece, in generale, relazioni di dominanza tra rilassamento per decomposizione e rilassamento surrogato.

Le relazioni di dominanza tra i vari rilassamenti del problema P (relativamente ai rispettivi moltiplicatori ottimi) possono dunque essere riassunte mediante il seguente schema:

$$\left[\begin{array}{c} Z(S_{(\mathrm{v})}(P,\pi^*)) \\ Z(D_{(\mathrm{v}),(\mathrm{w})}(P,\lambda^*)) \end{array} \right] \leq Z(L_{(\mathrm{v})}(P,\mu^*)) \leq \left[\begin{array}{c} Z(C(P)) \\ Z(E_{(\mathrm{v})}(P)) \end{array} \right].$$

5 Tecniche di Riduzione

In molte situazioni calcoli di upper bound (specialmente se richiedono uno sforzo computazionale relativamente modesto) possono consentire di ridurre la dimensione dell'istanza da risolvere. Si intende per *riduzione* l'operazione effettuata da un algoritmo in grado di stabilire preventivamente il valore che un certo numero di variabili debbono assumere in una soluzione ottima, consentendo quindi di applicare il successivo algoritmo risolvente ad un' istanza modificata e dipendente solo dalle rimanenti variabili.

Si consideri ancora il problema P (definito da (1)-(4)) introdotto nella Sezione 3 e si definisca il problema $P_{[x_h=1]}$ che si ottiene imponendo che la variabile x_h ($h \in \{1,\ldots,n\}$) assuma valore 1:

$$\begin{aligned} Z(P_{[x_h=1]}) = c_h + \quad \max \quad & \sum_{\substack{j=1 \\ j\neq h}}^{n} c_j x_j \\ & \sum_{\substack{j=1 \\ j\neq h}}^{n} a_{ij} x_j \leq b_i - a_{ih} \qquad (i = 1,\ldots,m), \\ & \sum_{\substack{j=1 \\ j\neq h}}^{n} d_{kj} x_j = e_k - d_{kh} \qquad (k = 1,\ldots,l), \\ & x_j \in \{0,1\} \qquad (j = 1,\ldots,n; j \neq h). \end{aligned}$$

Un rilassamento di questo problema fornirà un upper bound $U(P_{[x_h=1]})$ sul valore di $Z(P_{[x_h=1]})$. Sia Z il valore di una soluzione ammissibile di P (ottenuta, ad esempio, mediante un qualunque algoritmo approssimato). Evidentemente saremo interessati solo a soluzioni migliori di questa, per cui potremo applicare la seguente

Regola 1 *Se* $U(P_{[x_h=1]}) \leq Z$, *si deve avere* $x_h = 0$ *in qualunque soluzione di* P *di valore maggiore di* Z.

Definendo in maniera analoga $P_{[x_h=0]}$, si avrà anche, evidentemente:

Regola 2 *Se $U(P_{[x_h=0]}) \leq Z$, si deve avere $x_h = 1$ in qualunque soluzione di P di valore maggiore di Z.*

Calcolando $U(P_{[x_h=1]})$ ed $U(P_{[x_h=0]})$ per $h = 1, \ldots, n$ (o per un opportuno sottoinsieme di valori h) potremo quindi ridurre il problema definendo gli insiemi J_0 e J_1 contenenti le variabili che debbono assumere, rispettivamente, valore 0 e valore 1 in qualunque soluzione di P di valore maggiore di Z. Avremo quindi:

$$\begin{aligned} J_0 &= \{h : U(P_{[x_h=1]}) \leq Z\}; \\ J_1 &= \{h : U(P_{[x_h=0]}) \leq Z\}; \\ J &= \{1, \ldots, n\} \backslash (J_0 \bigcup J_1). \end{aligned}$$

Potranno a questo punto verificarsi alcuni casi particolari. Se $J_0 \bigcap J_1 \neq \emptyset$, potremo immediatamente concludere che imporre il miglioramento della soluzione di valore Z porta ad una contraddizione, cioè:

Regola 3 *Se $J_0 \bigcap J_1 \neq \emptyset$ la soluzione ammissibile di valore Z è ottima per P.*

Se $J = \emptyset$, avremo ottenuto una soluzione di valore $\sum_{h \in J_1} c_h$. Essa non sarà però necessariamente ammissibile (poichè J_0 e J_1 sono stati determinati mediante rilassamenti del problema), nè, se ammissibile, sarà necessariamente migliore di quella nota (poichè le Regole 1 e 2 danno condizioni necessarie ma non sufficienti per il miglioramento di Z). Pertanto

Regola 4 *Se $J = \emptyset$,*

(a) se $\sum_{h \in J_1} a_{ih} \leq b_i$ per $i = 1, \ldots, m$ e $\sum_{h \in J_1} d_{kh} = e_k$ per $k = 1, \ldots, l$, allora la soluzione ottima di P è la migliore tra quella definita da J_0 e J_1 (di valore $\sum_{h \in J_1} c_h$) e quella nota di valore Z;

(b) altrimenti la soluzione nota di valore Z è ottima per P.

Se invece $J_0 \bigcap J_1 = \emptyset$ e $J \neq \emptyset$, la soluzione di P potrà essere ottenuta risolvendo il problema ridotto $P_{[J_0, J_1]}$ indotto dagli insiemi J_0 e J_1:

$$Z(P_{[J_0,J_1]}) = \max \sum_{j \in J} c_j x_j$$

$$\sum_{j\in J} a_{ij}x_j \le b_i - \sum_{h\in J_1} a_{ih} \qquad (i = 1,\dots,m),$$
$$\sum_{j\in J} d_{kj}x_j = e_k - \sum_{h\in J_1} d_{kh} \qquad (k = 1,\dots,l),$$
$$x_j \in \{0,1\} \qquad (j \in J),$$

ed applicando la seguente regola (ottenuta in maniera analoga alla Regola 4):

Regola 5 *Se* $J_0 \bigcap J_1 = \emptyset$ *e* $J \neq \emptyset$*,*

(a) se $P_{[J_0,J_1]}$ *non ammette soluzione, allora la soluzione nota di valore* Z *è ottima per* P*;*

(b) altrimenti sia $J^* = \{j \in J : x_j = 1$ *nella soluzione ottima di* $P_{[J_0,J_1]}\}$*: la soluzione ottima di* P *è la migliore tra quella definita da* J_0*,* J_1 *e* J^* *(di valore* $\sum_{h\in J_1} c_h + \sum_{j\in J^*} c_j$*) e quella nota di valore* Z*.*

Bibliografia

T. Ibaraki (1987), *Enumerative Approaches to Combinatorial Optimization*, Annals of Operations Research 10-11, Baltzer, Basel.

G.L. Nemhauser and L.A. Wolsey (1988), *Integer and Combinatorial Optimization*, Wiley, Chichester.

S. Martello and P. Toth (1990), *Knapsack Problems: Algorithms and Computer Implementations*, Wiley, Chichester.

M. Minoux (1986), *Mathematical Programming: Theory and Algorithms*, Wiley, Chichester.

C.H. Papadimitriou and K. Steiglitz (1982), *Combinatorial Optimization: Algorithms and Complexity*, Prentice Hall, Engelwood Cliffs.

A. Schrijver (1986), *Theory of Linear and Integer Programming*, Wiley, Chichester.

J.F. Shapiro (1979), *Mathematical Programming: Structures and Algorithms*, Wiley, Chichester.