

Esercizi di Ricerca Operativa I

Dario Bauso, Raffaele Pesenti

Domande

- *Programmazione lineare intera*

1. Gli algoritmi per la programmazione lineare continua possono essere usati per la soluzione dei problemi di programmazione lineare binaria?
2. * Quando la formulazione di un problema di programmazione lineare intera si dice migliore di un'altra e perché?
3. Dimostrare che per ogni problema di programmazione lineare intera, con variabili vincolate superiormente, esiste una formulazione equivalente con solo variabili binarie.
4. Dato il seguente problema di programmazione lineare intera trasformarlo in problema di programmazione lineare binaria.

$$\begin{aligned}\max z &= +40x_a + 42x_b \\ 2x_a + x_b &\leq 6 \\ 7x_a + 3x_b &\leq 15 \\ 0 &\leq x_a \leq 4 \\ 0 &\leq x_b \leq 3.\end{aligned}$$

5. Dimostrare che le seguenti condizioni booleane su variabili binarie possono essere espresse con vincoli lineari: 1) $x_3 = x_1$ OR x_2 , 2) $x_3 = x_1$ AND x_2 , 3) $x_3 = NOT$ x_2 .
6. Perché è estremamente importante sapere risolvere problemi di programmazione lineare binaria?
7. Cosa è un metodo di enumerazione implicita per la soluzione di un problema di programmazione lineare binaria?
8. Quale è l'idea alla base del Branch & Bound?
9. L'algoritmo del Branch & Bound determina sempre la soluzione ottima di un problema di programmazione lineare binaria con un numero finito di variabili in un tempo finito?
10. Quale è il numero massimo dei nodi dell'albero di ricerca generato da un algoritmo di Branch & Bound applicato ad un problema di programmazione lineare binaria con n variabili?

11. Quale è la profondità massima dell'albero di ricerca generato da un algoritmo di Branch & Bound applicato ad un problema di programmazione lineare binaria con n variabili?
12. Stimare quale è il tempo di esecuzione massimo che l'algoritmo di Branch & Bound può richiedere per risolvere un problema con $n = 1000$ variabili binarie. Supporre di avere a disposizione un calcolatore da 10 teraflop e che il calcolatore sia capace di eseguire risolvere i problemi rilassati ad ogni nodo in sole 100 operazioni floating point. Approssimare per semplicità 2^{10} con 1000.
13. Stimare quale è il numero massimo di nodi che l'algoritmo di Branch & Bound può riuscire ad esplorare in 10.000 secondi (circa 2:45 ore). Indicare la dimensione massima, in termini di numero di variabili, dell'istanza di programmazione lineare binaria che può essere risolta in tale tempo nel caso in cui si il Branch & Bound debba generare un albero di ricerca completo. Supporre di avere a disposizione un calcolatore da 10 teraflop e che il calcolatore sia capace di eseguire risolvere i problemi rilassati ad ogni nodo in sole 100 operazioni floating point. Approssimare per semplicità 2^{10} con 1000.
14. Come mai, a volte, il Branch & Bound riesce a risolvere problemi di dimensioni significative in tempi ragionevoli?
15. Perché in molti software commerciali l'algoritmo del Branch & Bound si interrompe quando ha trovato una soluzione che è sicuramente entro il 5% (o l'1%) dall'ottimo?
16. Cosa è un metodo euristico di soluzione di un problema di programmazione lineare binaria?
17. Come ci si comporta se non si riesce a trovare la soluzione esatta di un problema di programmazione lineare intera in tempi ragionevoli, ad esempio attraverso un algoritmo di Branch & Bound?
18. Dato un algoritmo di Branch & Bound, cosa vuol dire che l'esplorazione dell'albero di ricerca avviene in modo depth first / best bound first?
19. Dato un algoritmo di Branch & Bound, quali sono i vantaggi e gli svantaggi legati alla visita dell'albero di ricerca in modo depth first / best bound first? (Dare per scontato che si sappia già cosa significa depth first / best bound first.)
20. Quale tra il depth first e il best bound first è utilizzato nella pratica dai software commerciali?
21. * Dato un problema di programmazione binaria, in che ordine vengono scelte le variabili che vengono progressivamente vincolate a valori interi nelle successive iterazioni del Branch & Bound?
22. Dato un problema di programmazione lineare binaria, cosa si intende per problema rilassato?
23. Quale è l'utilità di risolvere un problema rilassato invece che il programmazione lineare binaria originale?
24. * Quali sono i rilassamenti più utilizzati nella pratica per i problemi di programmazione lineare binaria?
25. * Cosa si intende per rilassamento continuo / per eliminazione / surrogato / lagrangiano?

26. * Sia dato un problema di massimo di programmazione lineare intera. Quando un rilassamento può dirsi migliore di un altro? Esiste un rilassamento ottimo per la classe dei rilassamenti lagrangiani? E per la classe dei rilassamenti surrogati?
27. * Quali relazioni di dominanza esistono tra i seguenti rilassamenti continuo / per eliminazione / surrogato / lagrangiano. Perché non si usa uno solo di essi?
28. Sia dato un problema di programmazione lineare binaria in cui bisogna MINIMIZZARE una data funzione obiettivo. Si risolvono tre rilassamenti diversi. Il rilassamento R_1 dà come soluzione $z_{R_1} = 150$, il rilassamento R_2 dà come soluzione $z_{R_2} = 140$, il rilassamento R_3 dà come soluzione $z_{R_3} = 120$. Quale dei tre rilassamenti si deve considerare migliore per l'istanza data?
29. * Sia dato il seguente problema di programmazione lineare binaria

$$\begin{aligned} \max z = & +40x_1 + 41x_2 + 42x_3 + 46x_4 + 47x_5 \\ & +42x_1 + 44x_2 + 47x_3 + 49x_4 + 53x_5 \leq 105 \\ & -7x_1 + 14x_2 + 17x_3 + 20x_4 + 22x_5 \leq 36 \\ & x_1, x_2, x_3, x_4, x_5 \in \{0, 1\}. \end{aligned}$$

Il suo rilassamento surrogato, in cui il primo vincolo è pesato 2 e il secondo 3, dà luogo alla soluzione $x_{RS} = (0, 0, 1, 0, 1)$ con $z_{RS} = 89$. Indicare qualitativamente come si dovrebbero cambiare i valori dei pesi per sperare di ottenere un rilassamento surrogato migliore.

30. * Dato il seguente problema di programmazione lineare binaria, sapendo che la soluzione ottima del suo rilassamento per eliminazione del primo vincolo è $x_{RE} = (0, 1, 1, 0, 1)$, valutare una stima dell'errore percentuale che si compie accettando la soluzione $\hat{x} = (1, 1, 1, 0, 0)$ piuttosto che cercare la soluzione ottima.

$$\begin{aligned} \max z = & +43x_1 + 45x_2 + 49x_3 + 53x_4 + 54x_5 \\ & +40x_1 + 43x_2 + 44x_3 + 47x_4 + 49x_5 \leq 127 \\ & +13x_1 + 14x_2 - 5x_3 + 18x_4 + 21x_5 \leq 33 \\ & x_1, x_2, x_3, x_4, x_5 \in \{0, 1\}. \end{aligned}$$

31. Dato un problema di programmazione lineare intera come può essere provata l'ottimalità di una soluzione? E la sua ammissibilità?
32. Cosa vuol dire che un problema è difficile e perchè è importante la Programmazione Lineare Intera nell'ambito di tali problemi.
33. Cos'è un problema di PLI?
34. In quali casi bisogna ricorrere alle variabili intere?
35. * Si definisca un problema di Ottimizzazione combinatoria.
36. Si definisca a parole il problema dello zaino (istanza, soluzione e obiettivo).

37. Si scriva il modello del problema dello zaino.
38. Citare e descrivere un esempio di applicazione del problema dello zaino.
39. Si definisca a parole il problema del bin packing (istanza, soluzione e obiettivo).
40. Si scriva il modello del problema del bin packing.
41. Citare e descrivere un esempio di applicazione del problema del bin packing.
42. * Si definisca a parole il problema dello scheduling (istanza, soluzione e obiettivo).
43. * Si scriva il modello del problema di scheduling.
44. Si definisca a parole il problema del mix di produzione con costi di set-up (istanza, soluzione e obiettivo).
45. Si scriva il modello del problema del mix di produzione con costi di set-up.
46. Si definisca a parole il problema del commesso viaggiatore (istanza, soluzione e obiettivo).
47. * Si scriva il modello del problema del commesso viaggiatore.
48. Si definisca una soluzione di tipo covering.
49. Si definisca una soluzione di tipo packing.
50. Si definisca una soluzione di tipo partitioning.
51. Si definisca un problema di covering.
52. Si definisca un problema di packing.
53. Si definisca un problema di partitioning.
54. * Si definisca la matrice e il vettore di incidenza e i vincoli di covering, packing e partitioning.

Risposte agli esercizi

Notazione

Nelle risposte successive

- $\max\{z = cx : Ax \leq b, x \in \{0, 1\}\}$ indica il generico problema binario

$$\begin{aligned} \max z &= cx \\ Ax &\leq b \\ x &\in \{0, 1\} \end{aligned}$$

- x^* indica la soluzione ottima del problema binario, e $z^* = cx^*$ il valore assunto all'ottimo dalla funzione obiettivo.
- x^*_{RC} indica la soluzione ottima del rilassamento continuo di un problema binario, e $z^*_{RC} = cx^*_{RC}$ il valore assunto all'ottimo dalla funzione obiettivo del rilassamento continuo.

Risposte

- (??) Affinché un problema di programmazione binaria $\max\{z = cx : Ax \leq b, x \in \{0, 1\}\}$ possa essere dato in input a un algoritmo per la programmazione lineare continua bisogna rilassare in vincoli di interezza e considerare il problema $\max\{z_{RC} = cx : Ax \leq b, 0 \leq x \leq 1\}$. In generale la soluzione ottima x_{RC}^* del problema rilassato risulta essere frazionaria. Vale inoltre la relazione $cx^* \leq cx_{RC}^*$, dove x^* è la soluzione ottima del problema binario. È infatti banale osservare che x^* è ammissibile per il problema rilassato e quindi che il valore ottimo del problema rilassato non può che essere migliore o al più uguale a quello ottimo del problema intero.

Gli algoritmi per la programmazione lineare continua possono essere utilizzati per determinare l'ottimo di un problema di programmazione intera nel caso in cui i vincoli del rilassamento continuo definiscono un poliedro che abbia come vertice ottimo una soluzione intera. Questa possibilità è sfruttata dagli algoritmi di soluzione in due modi differenti. 1) I metodi basati sui piani di taglio aggiungono successivamente vincoli al rilassamento continuo del problema originale in modo da eliminare progressivamente le soluzioni ottime ma frazionarie. Tali metodi iterano il loro procedimento fino a ottenere formulazioni che definiscono poliedri le con soluzioni ottime intere. 2) Il Branch & Bound suddivide ricorsivamente le soluzioni in sottopoliedri fino a quando tali sottopoliedri hanno vertici ottimi interi oppure certamente non contengono la soluzione ottima.

- (??) Una formulazione di un problema di programmazione lineare intera $\max\{z = cx : Ax \leq b, x \in \{0, 1\}\}$ si dice migliore della formulazione $\max\{z = cx : \hat{A}x \leq \hat{b}, x \in \{0, 1\}\}$ se le soluzioni ammissibili intere delle due formulazioni sono le stesse ma il poliedro $P = \{x : Ax \leq b, 0 \leq x \leq 1\}$ associato alla prima formulazione è incluso nel poliedro $\hat{P} = \{x : \hat{A}x \leq \hat{b}, 0 \leq x \leq 1\}$ associato alla seconda formulazione. In tale situazione è vero che $cx^* \leq cx_{RC}^* \leq c\hat{x}_{RC}^*$ per qualunque funzione di costo cx . Quindi, qualunque sia la funzione obiettivo, la soluzione del rilassamento continuo della prima formulazione fornisce una stima migliore del valore ottimo del problema di programmazione lineare intera rispetto al rilassamento continuo della seconda formulazione.
- (??) Sia dato un problema intero la cui generica variabile $x_i \in \mathbb{N}$ è limitata superiormente dal vincolo $0 \leq x_i \leq u_i$, dove u_i è una costante. Per ottenere il corrispondente problema binario basta sostituire nel problema originale la variabile x_i con la seguente combinazione di variabili binarie $\sum_{k=0}^{\lfloor \log_2 u_i \rfloor} 2^k y_{ki} = y_{0i} + 2y_{1i} + 4y_{2i} + \dots + 2^{\lfloor \log_2 u_i \rfloor} y_{\lfloor \log_2 u_i \rfloor, i}$. I valori assunti da ogni vettore di variabili $(y_{0i}, y_{1i}, y_{2i}, \dots, y_{\lfloor \log_2 u_i \rfloor, i})$ corrisponde all'espressione in codice binario del valore della variabile x_i .
- (??) Si pone $x_a = y_{0a} + 2y_{1a} + 4y_{2a}$ e $x_b = y_{0b} + 2y_{1b}$ ottenendo

$$\begin{aligned} \max z &= +40(y_{0a} + 2y_{1a} + 4y_{2a}) + 42(y_{0b} + 2y_{1b}) \\ 2(y_{0a} + 2y_{1a} + 4y_{2a}) + (y_{0b} + 2y_{1b}) &\leq 6 \\ 7(y_{0a} + 2y_{1a} + 4y_{2a}) + 3(y_{0b} + 2y_{1b}) &\leq 15 \\ 0 \leq y_{0a} + 2y_{1a} + 4y_{2a} &\leq 4 \end{aligned}$$

$$0 \leq y_{0b} + 2y_{1b} \leq 3,$$

da cui

$$\begin{aligned} \max z &= +40y_{0a} + 80y_{1a} + 160y_{2a} + 42y_{0b} + 84y_{1b} \\ 2y_{0a} + 4y_{1a} + 16y_{2a} + y_{0b} + 2y_{1b} &\leq 6 \\ 7y_{0a} + 14y_{1a} + 28y_{2a} + 3y_{0b} + 6y_{1b} &\leq 15 \\ 0 \leq y_{0a} + 2y_{1a} + 4y_{2a} &\leq 4 \\ 0 \leq y_{0b} + 2y_{1b} &\leq 3. \end{aligned}$$

- (??) 1) La condizione $x_3 = x_1$ OR x_2 può essere espressa dall'insieme delle seguenti condizioni

$$\begin{cases} x_3 \leq x_1 + x_2 \\ x_3 \geq x_1 \\ x_3 \geq x_2 \end{cases}$$
- 2) La condizione $x_3 = x_1$ AND x_2 può essere espressa dall'insieme delle seguenti condizioni

$$\begin{cases} x_3 \geq x_1 + x_2 - 1 \\ x_3 \leq x_1 \\ x_3 \leq x_2 \end{cases}$$
- 3) La condizione $x_3 = NOT$ x_2 può essere espressa come $x_3 = 1 - x_2$.
- (??) È estremamente importante sapere risolvere problemi di programmazione lineare binaria perché, ad esempio, tramite di essi si possono modellare tutte quelle situazioni decisionali descrivibili attraverso la logica del primo ordine. Più tecnicamente si possono risolvere tutti i problemi della classe NP. (Gli interessati possono vedere le dispense corso di Ricerca Operativa II sulla complessità computazionale.)
- (??) Un metodo è detto di enumerazione implicita se determina la soluzione ottima di un problema esplorando in modo esplicito o implicito tutte le soluzioni del problema stesso. In particolare si definisce esplorazione esplicita di una soluzione \hat{x} il calcolo del corrispondente valore $c\hat{x}$. Viceversa si definisce esplorazione implicita di una classe di soluzioni l'escludere a priori che a tutti, o ad almeno alcuni, elementi della classe sia associato il valore ottimo di cx senza la necessità di enumerare esplicitamente le soluzioni considerate.
- (??) Il Branch & Bound è un metodo di enumerazione implicita delle soluzioni. Il Branch & Bound suddivide ricorsivamente l'insieme delle soluzioni di un problema in sottoinsiemi. La suddivisione si ferma a quando per ognuno dei sottoinsiemi si riesce a valutare il valore cx della migliore soluzione inclusa nel sottoinsieme oppure si può escludere che il sottoinsieme contenga la soluzione ottima. I sottoinsiemi generati possono essere organizzati ad albero. Ogni sottoinsieme generato corrisponde ad un nodo dell'albero. Ad ogni iterazione il Branch & Bound analizza il sottoinsieme di soluzioni associato ad un nodo dell'albero di ricerca e determina una stima \hat{z} del valore cx della migliore soluzione del sottoinsieme. Se \hat{z} non esiste perché tutte le soluzioni non sono ammissibili o \hat{z} risulta peggiore della migliore soluzione intera correntemente disponibile all'algoritmo, il Branch & Bound esclude da

ulteriore analisi le soluzioni del sottoinsieme considerato. In questo caso si dice che il Branch & Bound pota il nodo dell'albero di ricerca.

- (??) L'algoritmo del Branch & Bound determina sempre la soluzione ottima di un problema di programmazione lineare binaria con un numero finito di variabili in un tempo finito. Infatti, per svolgere la ricerca dell'ottimo, l'algoritmo produce un albero con un numero finito di nodi e, in ogni nodo, risolve un problema di programmazione lineare continua in un tempo finito.
- (??) L'albero di ricerca generato da un algoritmo di Branch & Bound applicato a un problema di programmazione lineare binaria con n variabili contiene al più $2^{n+1} - 1$ nodi.
- (??) La profondità massima dell'albero di ricerca generato da un algoritmo di Branch & Bound applicato ad un problema di programmazione lineare binaria con n variabili è di n livelli. Ad ogni livello infatti è fissato il valore di una delle n variabili.
- (??) Nel caso peggiore l'algoritmo di Branch & Bound deve esplorare tutti i $2^{n+1} - 1$ nodi dell'albero di ricerca. Siccome in ogni nodo il Branch & Bound deve risolvere un problema rilassato si ottiene che nel caso peggiore devono essere eseguite $(2^{1001} - 1) \times 100 \approx 10^{300} \times 100 = 10^{302}$ operazioni floating point. Un calcolatore da 10 teraflop, cioè da 10^{13} operazioni al secondo, impiegherebbe $10^{302}/10^{13} = 10^{289}$ secondi. Per avere un termine di paragone si tenga presente che si suppone che l'età dell'universo sia di 10^{17} secondi.
- (??) Un calcolatore da 10 teraflop esegue 10^{13} operazioni al secondo. Se per risolvere un problema rilassato servono 100 operazioni, l'algoritmo di Branch & Bound può esplorare 10^{11} nodi al secondo e quindi 10^{15} nodi in 10.000 secondi. Un albero di ricerca associato ad un'istanza di n variabili binarie contiene al più $2^{n+1} - 1$ nodi. Ponendo $2^{n+1} - 1 = 10^{15}$ si ottiene $n \approx 50$.
- (??) A volte il Branch & Bound riesce a risolvere problemi anche con numerose variabili in tempi ragionevoli in quanto riesce ad escludere a priori sottoinsiemi di soluzioni molto grandi. Le prestazioni dell'algoritmo dipendono in maniera drammatica dal livello a cui avvengono le potature dei nodi dell'albero di enumerazione. È fondamentale quindi che i problemi rilassati che vengono risolti ad ogni nodo: risultino quanto più spesso possibile inammissibili se il corrispondente problema intero non ammette soluzioni ammissibili oppure forniscano un bound sul valore ottimo il più vicino possibile a quello del problema intero.
- (??) I problemi di programmazione lineari interi o binari spesso presentano moltissime soluzioni ottime equivalenti. Molti software commerciali che implementano il Branch & Bound si interrompono quando hanno trovato una soluzione che è sicuramente entro il 5% (o l'1%) dall'ottimo per proprio per evitare di enumerare tutte queste soluzioni.
- (??) I metodi euristici di soluzione sono algoritmi che risolvono i problemi di ottimizzazione. Essi utilizzano in genere regole di buon senso e restituiscono soluzioni ammissibili ma non necessariamente

ottime. I metodo euristici forniscono quindi soluzioni peggiori dei metodi di enumerazione implicita che restituiscono soluzioni ottime. I metodi euristici però terminano tipicamente in un numero di passi molto inferiore a quelli degli algoritmi di enumerazione.

- (??) Dato un problema programmazione lineare intera se non si riesce a trovare la soluzione ottima in tempi ragionevoli, ad esempio attraverso un algoritmo di Branch & Bound, si passa a qualche metodo euristico. In questo caso si sacrifica la qualità della soluzione in cambio di una risposta veloce.
- (??) Dato un algoritmo di Branch & Bound, si dice che l'esplorazione dell'albero di ricerca avviene in modo depth first se ad ogni iterazione l'algoritmo sceglie di esplorare uno tra i nodi figli del nodo appena esplorato. Se il nodo appena esplorato è stato potato o non ha figli, l'algoritmo passa al nodo di più basso livello non ancora esplorato. Si dice che l'esplorazione dell'albero di ricerca avviene in modo best bound first se ad ogni iterazione l'algoritmo sceglie di esplorare uno tra i nodi a cui è associato il migliore bound della soluzione ottima.
- (??) Dato un algoritmo di Branch & Bound, i vantaggi del depth first sono legati al fatto che ad ogni iterazione devono essere memorizzati al più n nodi già esplorati e n nodi in attesa di essere esplorati. Gli svantaggi sono legati al fatto che ad ogni iterazione si può scegliere solo tra un numero molto limitato di nodi e quindi le soluzioni che si trovano all'inizio dell'esplorazione possono essere molto scadenti. I vantaggi del best bound first sono dovuti al fatto che la prima soluzione intera che si determina con questa strategia di ricerca o è ottima o, in generale, è molto vicina ad essere tale. Gli svantaggi è che il numero dei nodi che rimangono aperti ad ogni iterazione può crescere notevolmente e occupare tutto lo spazio di memoria disponibile.
- (??) I software commerciali tipicamente usano una strategia mista tra depth first e best bound first. Essi applicano una strategia depth first fino a quando i bound dei nodi figli dell'ultimo nodo esplorato non risultano scadenti. Quando ciò accade, abbandonano il nodo corrente e ripartono con una strategia depth first dal nodo aperto con il migliore bound.
- (??) Quando è possibile è meglio scegliere prima le variabili più importanti, cioè quelle variabili in cui imporre un valore intero implica dei vincoli molto stringenti (possibilmente addirittura l'interezza) sul maggior numero delle rimanenti variabili non ancora fissate.
- (??) Dato un problema di programmazione lineare binaria, si definisce rilassamento un problema tale che: 1) ha un insieme di soluzioni ammissibili che include le soluzioni del problema originale, 2) ha una funzione obiettivo il cui valore è sempre non peggiore di quello della funzione obiettivo del problema originale quando tali funzioni sono calcolate rispetto a soluzioni ammissibili per entrambe i problemi. Tipicamente un problema rilassato si ottiene rendendo più deboli alcuni dei vincoli del problema originale. Ad esempio nel rilassamento continuo si rinuncia all'interezza delle variabili.
- (??) I problemi rilassati sono, in generale, più semplici da risolvere rispetto al problema originale.

La soluzione ottima del problema rilassato permette di determinare una stima del valore ottimo del problema originale, anche se di norma tale soluzione non è ammissibile per il problema originale.

- (??) Il rilassamento lineare (o continuo) e il rilassamento lagrangiano.
- (??) Dato un problema di programmazione binaria, nel rilassamento continuo viene a mancare il vincolo di interezza che viene sostituito dal vincolo $0 \leq x_i \leq 1$ per ogni variabile binaria nel problema originale. Nel rilassamento per eliminazione vengono eliminati alcuni dei vincoli (ma non quello di interezza). Nel rilassamento surrogato alcuni gruppi di vincoli vengono sostituiti da loro combinazioni coniche. Nel rilassamento lagrangiano alcuni dei vincoli vengono trasformati in componenti pesate dell'obiettivo.
- (??) Un rilassamento può dirsi migliore di un altro quando fornisce una stima migliore del valore ottimo del problema originale. Nel caso di problemi di massimizzazione il valore ottimo di un problema rilassato z_R è sempre maggiore del valore ottimo z^* del problema originale. Un rilassamento R è quindi migliore di un altro rilassamento \hat{R} se $z_{\hat{R}} \geq z_R \geq z^*$. Il valore di z_R è infatti una migliore stima di z^* .

Il valore del ottimo del rilassamento lagrangiano z_{RL} dipende dai pesi che sono stati utilizzati per rilassare i vincoli. Il rilassamento lagrangiano surrogato ottimo è quindi quello per cui i pesi scelti rendono minimo il valore z_{RL} . Il lagrangiano surrogato ottimo viene detto duale lagrangiano. Identico discorso vale per il rilassamento surrogato.

- (??) Il rilassamento surrogato ottimo è non peggiore del rilassamento lagrangiano ottimo che, a sua volta, è non peggiore sia del rilassamento lineare che di quello per eliminazione. Non esiste relazione di dominanza fra gli ultimi due rilassamenti. Non si usa solo un tipo di rilassamento perché alcuni di essi sono più facili da risolvere, altri danno stime migliori del valore ottimo del problema originale.
- (??) Trovandosi a risolvere un problema di minimizzazione vale che qualunque rilassamento assume un valore inferiore o uguale all'ottimo e quindi $z_{R_3} \leq z_{R_2} \leq z_{R_1} \leq z^*$. Poiché $|z^* - z_{R_1}| \leq |z^* - z_{R_2}| \leq |z^* - z_{R_3}|$, ne consegue che z_{R_1} è una migliore stima di z^* , da cui R_1 deve considerarsi il migliore rilassamento per il caso in questione.

- (??) Dato

$$\begin{aligned} \max z &= +40x_1 + 41x_2 + 42x_3 + 46x_4 + 47x_5 \\ +42x_1 + 44x_2 + 47x_3 + 49x_4 + 53x_5 &\leq 105 \\ -7x_1 + 14x_2 + 17x_3 + 20x_4 + 22x_5 &\leq 36 \\ x_1, x_2, x_3, x_4, x_5 &\in \{0, 1\} \end{aligned}$$

Si osserva che la soluzione del rilassamento surrogato $x_{RS} = (0, 0, 1, 0, 1)$ soddisfa il primo vincolo ma non il secondo. Per sperare di ottenere un rilassamento surrogato migliore converrebbe provare ad aumentare il peso assegnato al secondo vincolo e diminuire il peso assegnato al primo vincolo.

- (??) Il valore della funzione obiettivo del rilassamento per eliminazione è $z_{RE} = 148$. Il valore della funzione obiettivo associato alla soluzione ammissibile è $\hat{z} = 137$. L'errore percentuale che si compie accettando \hat{x} è $e\% = \frac{z^* - \hat{z}}{\hat{z}} \leq \frac{z_{RE} - \hat{z}}{\hat{z}} = \frac{11}{137} \approx 8\%$.
- (??) In generale non si può provare l'ottimalità di una soluzione di un problema di programmazione lineare binaria. Al più si può stimare per eccesso la sua distanza dall'ottimo facendo uso dei risultati di un rilassamento. Verificare l'ammissibilità di una soluzione è invece banale, basta controllare che tale soluzione soddisfi tutti i vincoli del problema.
- (??) I problemi difficili sono quelli per i quali è molto improbabile che si trovino algoritmi polinomiali esatti di soluzione. La PLI rappresenta un problema di riferimento a cui possibilmente trasformare o ridurre ogni altro problema difficile (NP-hard).
- (??) È un problema di ottimizzazione in cui:
 - La funzione obiettivo c è lineare
 - le condizioni che descrivono l'insieme delle soluzioni ammissibili sono lineari
 - le soluzioni ammissibili sono intere
 - i parametri sono deterministici
- (??) i) Per modellare leve decisionali intrinsecamente intere (macchine, persone...); ii) per esprimere relazioni logiche *OR*, *IF...THEN* tra le variabili; iii) per modellare scelte mutualmente esclusive (vero/falso, appartenenza/non appartenenza)
- (??) Sono dati un insieme finito $N = 1 \dots, N$ e un vettore $c^T = [c_1, \dots, c_n]$. Dato un generico sottoinsieme $F \subseteq N$, si definisce $c(F)$ come $c(F) = \sum_{i \in F} c_i$. Sia data una collezione S di sottoinsiemi di N , allora un problema di ottimizzazione combinatoria è dato da $\max_{F \in S} c(F)$.
- (??) *Istanza*: un insieme finito U , dove ogni $i \in U$ è caratterizzato da un volume $s_i \in N$, da un valore $v_i \in N$. Inoltre sia dato il volume dello zaino $B \in N$. *Soluzione*: un sottoinsieme $U' \subseteq U$ tale che il volume occupato non ecceda il volume dello zaino, i.e., $\sum_{i \in U'} s_i \leq B$. *Obiettivo*: il valore totale degli elementi scelti sia massimo, $\max_{U'} \sum_{i \in U'} c_i$.
- (??) Il modello del problema dello zaino è il seguente

$$\begin{aligned} \max \quad & \sum_{i=1}^n c_i x_i \\ \text{s.t.} \quad & \sum_{i=1}^n s_i x_i \leq B \\ & x_i = \{0, 1\}, \quad i = 1 \dots, n \end{aligned}$$
- (??) Capital Budgeting. Obiettivo: max ritorno atteso; vincoli: non investire più capitale di quello a disposizione; leve decisionali: scelta degli investimenti; dati tecnologici: p_i ritorno investimento i mo, a_i capitale richiesto investimento i mo, b capitale disponibile.

- (??) *Istanza*: un insieme finito U , dove ogni $u \in U$ è caratterizzato da un volume $s_u \in N$, e dei contenitori di volume $B \in N$. *Soluzione*: una partizione di U in sottoinsiemi $U(i)$, $i = 1, \dots, n$ ognuno dei quali assegnato a un bin, e tale che il volume occupato in ogni bin i non sia maggiore di B , i.e., $\sum_{u \in U(i)} s_u \leq B$. *Obiettivo*: il numero totale di bin sia minimo.

- (??) Il modello del problema del bin packing è il seguente

$$\begin{aligned} \min \sum_{i=1}^n y_i \\ \sum_{j=1}^n s_j x_{ij} \leq B y_i \quad \forall i = 1, \dots, n \\ \sum_{i=1}^n x_{ij} = 1 \quad \forall j = 1, \dots, n \\ x_{ij} \in \{0, 1\}, \quad \forall i, j = 1, \dots, n \end{aligned}$$

- (??) *Schedulazione operazioni di ortopedia*. Obiettivo: minimizzare numero di giorni utilizzati; vincoli: ogni operazione deve essere eseguita, la giornata lavorativa i ma non deve durare più di B_i ore; leve decisionali: quali giorni dedicare alle operazioni, quali operazioni effettuare in un dato giorno; dati tecnologici: a_j durata operazione j ma, B_i tempo disponibile per operare il giorno i mo.

- (??) *Istanza*: un'unica macchina e un insieme finito di operazioni U , dove ogni operazione $i \in U$ è caratterizzata da un tempo di esecuzione $p_i \in N$ e un tempo di rilascio $r_i \in N$. *Soluzione*: una sequenziazione delle operazioni in modo che nessuna operazione sia venga iniziata prima del proprio tempo di rilascio, venga eseguita contemporaneamente a un'altra. *Obiettivo*: il tempo medio di completamento delle operazioni sia minimo, i.e., $\min \sum C_i$.

- (??) Il modello del problema di scheduling è il seguente ($x_{ij} = 1$ se i precede j e 0 altrimenti)

$$\begin{aligned} \min \sum_{i=1}^n C_i \\ C_i \geq r_i + p_i \quad \forall i = 1, \dots, n \\ C_i \geq C_j + p_i - M x_{ij} \quad \forall i, j = 1, \dots, n, i \neq j \\ x_{ij} \in \{0, 1\}, \quad \forall i, j = 1, \dots, n \end{aligned}$$

- (??) *Istanza*: un insieme finito di tipologie di prodotti U e un insieme di risorse R , dove ogni unità di prodotto $i \in U$ è caratterizzato da un profitto $c_i \in N$, da un consumo a_{ji} di risorsa j , e da un costo fisso di inizio produzione d_i . *Soluzione*: un mix di produzione x che non richieda più risorse di quelle disponibili, i.e., $\sum_{i=1}^n a_{ji} x_i \leq b_j, \forall j = 1, \dots, m$, *Obiettivo*: il profitto del mix di produzione sia massimo al netto dei costi di set-up.

- (??) Il modello del problema del mix di produzione è il seguente

$$\max \sum_{i \in U} (c_i x_i - d_i y_i)$$

$$\begin{aligned} \sum_{i \in U} a_{ji} x_i &\leq b_j \quad \forall j \in R \\ x_i &\leq M y_i \quad \forall i \in U \\ x_i &\geq 0, y_i = \{0, 1\}, \quad \forall i \in U. \end{aligned}$$

- (??) *Istanza*: una rete completa $G(V, E)$ in cui a ogni arco e è associato un costo c_e di attraversamento, un nodo origine $s \in V$. *Soluzione*: un circuito $C = \{s, \dots, i, \dots, s\}$, che visiti tutti i nodi una sola volta, si richiuda sul nodo origine s e rispetti gli eventuali orientamenti degli archi. *Obiettivo*: il costo totale del circuito sia minimo, i.e., detto $E(C)$ l'insieme degli archi attraversati, $\min_{C: \text{circuiti}} \sum_{e \in E(C)} c_e$.

- (??) modello del problema del commesso viaggiatore è il seguente

$$\begin{aligned} \min \sum_{e \in E} c_e x_e \\ \sum_{e \in \delta(i)} x_e &= 2 \quad \forall i \in V \\ \sum_{e \in E(S)} x_e &\leq |S| - 1 \quad \forall S \subset V \\ x_e &\in \{0, 1\}, \quad \forall e \in E. \end{aligned}$$

- (??) Siano dati un insieme finito $M = \{1, \dots, m\}$, un insieme $S = \{F_1, \dots, F_n\}$ dei sottoinsiemi di M e un insieme $F = \{F_1, \dots, F_r\}$ di sottoinsiemi di M tali che $F \subseteq S$. L'insieme F si dice *covering* se $\bigcup_i F_i = M$.
- (??) Siano dati un insieme finito $M = \{1, \dots, m\}$, un insieme $S = \{F_1, \dots, F_n\}$ dei sottoinsiemi di M e un insieme $F = \{F_1, \dots, F_r\}$ di sottoinsiemi di M tali che $F \subseteq S$. L'insieme F si dice *packing* se $F_i \cap F_j = \emptyset$ for all $F_i, F_j \in F$.
- (??) Siano dati un insieme finito $M = \{1, \dots, m\}$, un insieme $S = \{F_1, \dots, F_n\}$ dei sottoinsiemi di M e un insieme $F = \{F_1, \dots, F_r\}$ di sottoinsiemi di M tali che $F \subseteq S$. L'insieme F si dice *partitioning* se $\bigcup_i F_i = M$ e $F_i \cap F_j = \emptyset$ for all $F_i, F_j \in F$.
- (??) *Istanza*: dati un insieme finito $M = \{1, \dots, m\}$ e una funzione $c: S \rightarrow N$ che assegni un costo a ogni sottoinsieme F_i di M . *Soluzione*: un covering $F = \{F_1, \dots, F_r\}$. *Obiettivo*: il costo del covering sia minimo, i.e., $\min_{F \subseteq S} \sum_{F_i \in F} c(F_i)$.
- (??) *Istanza*: dati un insieme finito $M = \{1, \dots, m\}$ e una funzione $c: S \rightarrow N$ che assegni un costo a ogni sottoinsieme F_i di M . *Soluzione*: un packing $F = \{F_1, \dots, F_r\}$. *Obiettivo*: il costo del packing sia massimo, i.e., $\max_{F \subseteq S} \sum_{F_i \in F} c(F_i)$.
- (??) *Istanza*: dati un insieme finito $M = \{1, \dots, m\}$ e una funzione $c: S \rightarrow N$ che assegni un costo a ogni sottoinsieme F_i di M . *Soluzione*: un partitioning $F = \{F_1, \dots, F_r\}$. *Obiettivo*: il costo del partitioning sia minimo, i.e., $\min_{F \subseteq S} \sum_{F_i \in F} c(F_i)$.

- (??) Matrice di incidenza $A \in \mathbb{R}^{m \times n}$ (una riga per ogni elemento di M e una colonna per ogni elemento F_j di S):

$$a_{ij} = \begin{cases} 1 & \text{se } i \in F_j \\ 0 & \text{altrimenti} \end{cases}$$

Vettore di incidenza $x \in \mathbb{R}^{n \times 1}$ (una riga per ogni elemento F_j di S):

$$x_i = \begin{cases} 1 & \text{se viene scelto } F_i \\ 0 & \text{altrimenti} \end{cases}$$

Covering: $Ax \geq 1$; Packing: $Ax \leq 1$; Partitioning: $Ax = 1$.