26-01-2017

Immagine che contiene tavolo

Descrizione generata automaticamente

Cominciamo con il creare le variabili decisionali per modellare i bavaglini in base al colore e le confezioni in base al colore come segue:

: quantità di bavaglini del tipo

: metri di filato di bavaglini della confezione

Quindi, volendo massimizzare i ricavi di beneficenza, avremo che:

* Ogni bavaglino viene venduto a 5 euro
* Ci sono dei costi di produzione, che vanno *sottratti* dai ricavi

(vincolo tempo complessivo, sapendo che per fare un bavaglino ci si impiegano 15 minuti e abbiamo convertito le 200 ore in minuti, quindi diventerebbe )

Sappiamo inoltre che:

* “sono richiesti almeno 10 bavaglini per tipo”
* “si vogliono acquistare al massimo due tipi di confezione”

Ciò richiede la creazione di un’apposita variabile binaria.

: variabile logica che vale 1 se acquistiamo la confezione del tipo

Con un vincolo di attivazione ,

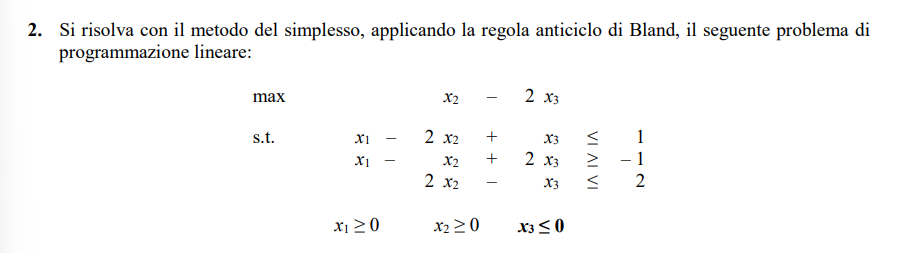
* “ciascun fornitore pratica uno sconto del 5% sul prezzo unitario di vendita se si acquistano almeno 10 delle loro confezioni (suggerisce di modellare la decisione sul n. di conf. da acquistare scontate)

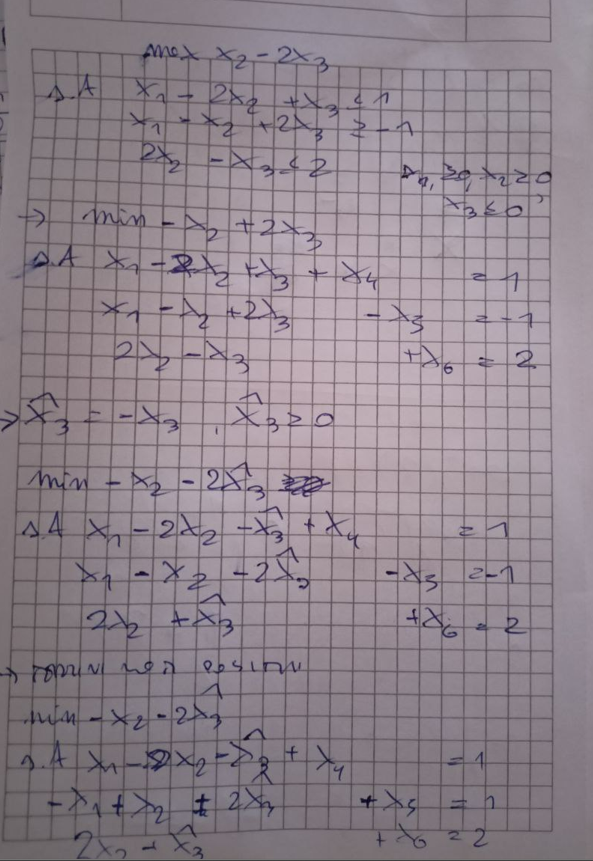
Usiamo una variabile logica per :

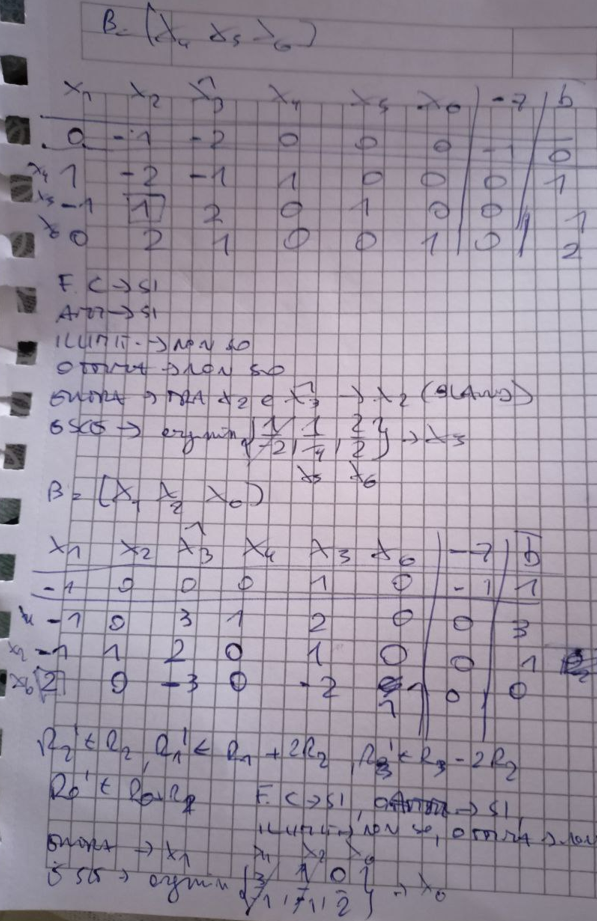
: variabile logica che vale 1 se acquistiamo a prezzo scontato confezioni del tipo , 0 altrimenti

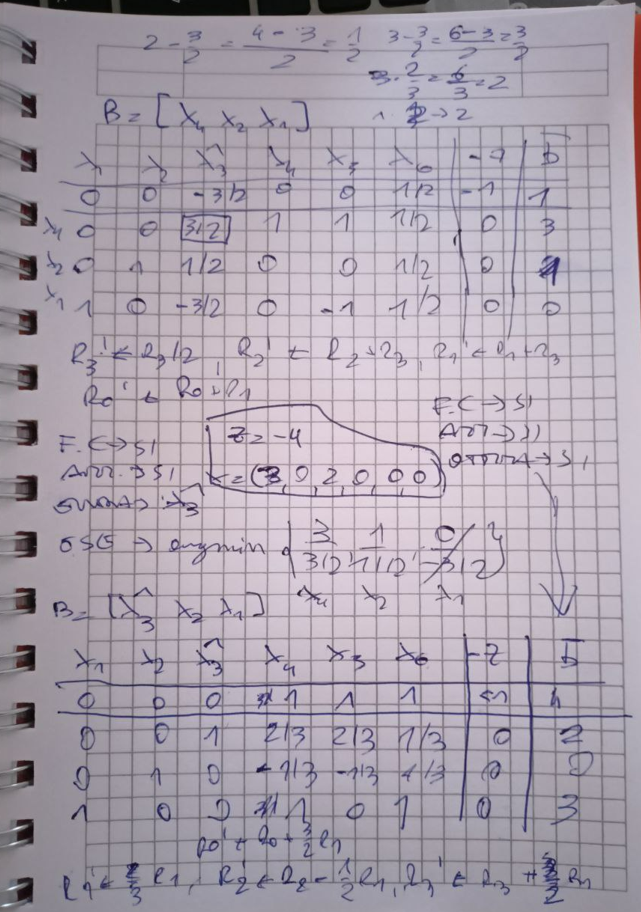
Implicitamente, si considerano i seguenti vincoli spuri, considerando lo sconto del 5% in funzione delle variabili a prezzo pieno presenti:

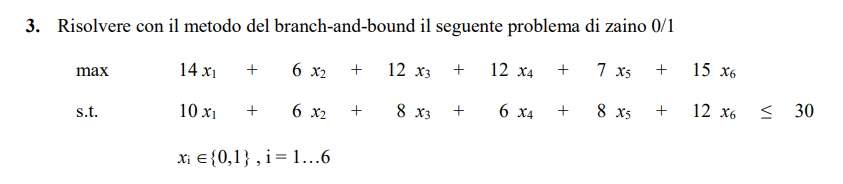
Domini:





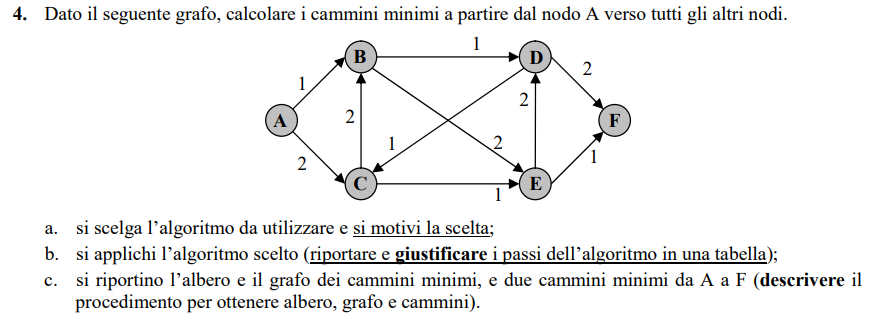


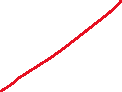




Per risolvere questo problema knapsack 0/1 utilizzando il metodo branch and bound, possiamo procedere come segue:

* Si inizia creando un elenco di elementi, dove ogni elemento è rappresentato come una tupla (valore, peso). Per il problema dato, l'elenco di elementi sarebbe:
* [(14, 10), (6, 6), (12, 8), (12, 6), (7, 8), (15, 12)]
* Ordinare l'elenco degli elementi in ordine non crescente rispetto al rapporto tra valore e peso. Questo ci permetterà di considerare prima gli elementi di maggior valore.
* Creare una funzione upper bound che stimi il valore massimo ottenibile dagli elementi rimanenti. Un modo semplice per farlo è includere tutti gli oggetti rimanenti nello zaino, indipendentemente dal loro peso. In questo modo si otterrà un upper bound sul valore totale che si può ottenere.
* Creare una funzione che esegua la ricerca branch and bound. Questa funzione deve avere come input l'elenco degli elementi, il peso corrente, il valore corrente e il limite superiore.
* Inizializzare il miglior valore trovato finora a zero.
* Se il peso corrente è maggiore del peso massimo, restituisce il miglior valore trovato finora.
* Se il valore corrente più il limite superiore è inferiore al valore migliore trovato finora, restituisce il valore migliore trovato finora.
* Se non ci sono altri elementi da considerare, restituisce il valore attuale.
* Per ogni elemento, calcolare il valore e il peso se l'elemento è incluso nello zaino e il valore e il peso se l'elemento non è incluso nello zaino.
* Richiamare ricorsivamente la funzione branch and bound con il valore e il peso aggiornati per entrambi i casi e aggiornare il miglior valore trovato finora, se necessario.
* Restituire il miglior valore trovato finora.





a) Il miglior algoritmo in questo contesto è Dijkstra, in quanto più efficiente e usabile in quanto tutti i costi ridotti sono positivi.

b) Ora, il calcolo dei cammini minimi, riportati in tabella. Si ricorda che:

* rappresenta l’etichetta minima di ogni iterazione
* rappresentano le etichette ancora da fissare
* Il segno \* rappresenta l’etichetta fissata
* Il segno – rappresenta l’etichetta controllata ma non aggiornata
* Il segno x rappresenta l’etichetta non controllata perché il nodo è già fissato
* Gli spazi vuoti servono per indicare che non considero più l’etichetta nelle varie iterazioni in quanto fissata
* L’algoritmo termina quando non ci sono più nodi in

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Iterazione | Nodo | Nodo | Nodo | Nodo | Nodo | Nodo |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  | \* |  |  |  |  |  |  |
|  |  |  |  | \* |  |  |  |  |
|  |  |  | \* |  |  |  |  |  |
|  |  |  |  |  | \* |  |  |  |
|  |  |  |  |  |  | \* |  |  |

Ad ogni iterazione, percorriamo il grafo e scegliamo il percorso con costo minore. Ad ogni iterazione, scegliamo e fissiamo un’etichetta che ha costo minore, scremando ad ogni iterazione quelle da controllare e avere sempre in mano il costo minimo. Anche in questo caso, come per gli algoritmi label correcting in caso di convergenza alla soluzione ottima, le etichette calcolate e i relativi puntatori rappresentano, rispettivamente, una soluzione duale ammissibile e i predecessori su dei cammini dall’origine ai diversi nodi. Questo funziona non avendo costi negativi.

c) Come abbiamo visto per gli algoritmi label correcting in caso di convergenza, è possibile derivare l’albero (risp. il grafo) dei cammini minimi attraverso i puntatori ai predecessori (risp. la verifica della saturazione dei vincoli duali sugli archi).

Contrariamente a quanto chiede la domanda, non esistono altri cammini minimi, seguendo tutti i possibili percorsi. Albero e grafo coincidono.

Immagine che contiene tavolo

Descrizione generata automaticamente

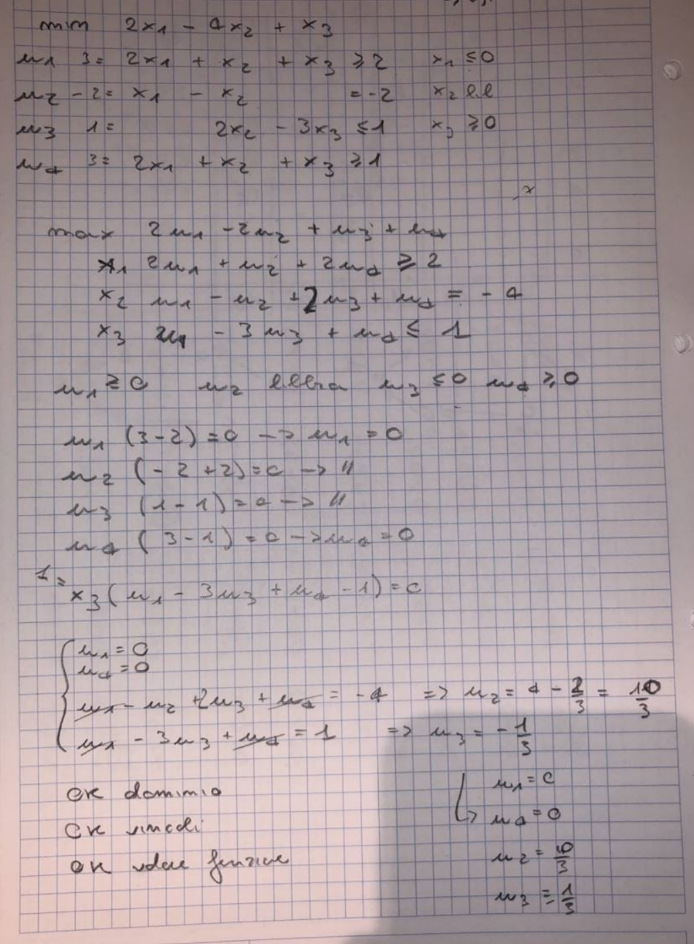


Immagine che contiene tavolo

Descrizione generata automaticamente

a) La soluzione di base corrispondente è data dall’individuazione dei pivot della matrice identità, quindi avremo oppure . Mentre la prima non è ottima, la seconda non si sa se lo sia, in quanto i coefficienti sono tutti ma non sono tutti positivi.

b) Non è consentita l’operazione su quell’elemento in quanto non rispetta la regola di individuazione dell’elemento di pivot rispetto a variabile che entra/variabile che esce. Si dovrebbe scegliere per Bland ed effettuare lì il rapporto minimo. Qui tale cosa non accade, dunque, non viene rispettata la regola del rapporto minimo.

c) Per le regole del simplesso, possiamo effettuare pivot sulle coppie di elementi e

d) Il cambio base applicando Bland viene dato su . In questo caso, avremmo due righe corrispondenti al rapporto minimo, pertanto avremo che almeno una esce dalla base con valore 0, mentre le altre due assumeranno valore 0 rimanendo in base.