Numero minimo di verghe

Carlo, ha un amico che lavora in un negozio di edilizia che spesso i trova a dover soddisfare richieste del tipo:" mi prepari n1 pezzi di verga metallica di lunghezza l1, n2 pezzi di verga di lunghezza l2, ...,nk di lunghezza lk". Naturalmente questi pezzi vengono tagliati da verghe di lunghezza standard, diciamo L (ad es. 600 cm) e in generale l1, l2, ...,lk, che sono minori di L, non sono tutti sottomultipli di L e, qualora alcuni di essi lo siano, il numero di pezzi di lunghezza li ricavabili da una verga (L/li) potrebbe non essere sottomultiplo di ni, come pure il numero di pezzi di lunghezza lj ricavabili da una verga potrebbe non esserlo di nj, e così via .
Il suo problema consiste nel trovare quante verghe occorrono per soddisfare le richieste, in modo da ridurre gli scarti al minimo e quindi, ovviamente, i costi.

Siano **L** la lunghezza standard delle verghe disponibili e **N** il numero di verghe strettamente necessario per soddisfare la richiesta.

Qualunque sia la composizione della richiesta, N non potrà mai scendere al disotto di N_1 = numero di verghe di lunghezza L necessario a coprire la somma delle lunghezze di tutti i pezzi richiesti, dato da

$$N_1 = \mathrm{INT}\left(\frac{\sum n_h l_h}{L}\right) + 1 \qquad h = 1, 2,3 \ldots. K \qquad \text{se} \quad \sum n_h\, l_h \quad \mathrm{MOD}\ \ L \neq 0$$

$$N_1 = \mathrm{INT}\left(\frac{\sum n_h l_h}{L}\right) \qquad h = 1, 2,3 \ldots. K \qquad \text{se} \quad \sum n_h\, l_h \quad \mathrm{MOD}\ \ L = 0$$

Il problema di della ripartizione ottimale dei pezzi richiesti tra varie verghe può essere affrontato con il seguente algoritmo:

Sia N_2 il numero totale dei pezzi richiesti

$$N_2 = \sum n_h \qquad\qquad h = 1, 2,3 \ldots. K$$

N_2 costituisce il massimo valore possibile per N, che si verifica quando tutti i pezzi richiesti hanno lunghezza superiore a L/2.

Costruiamo una sequenza $S = \{ l_1, l_1, \ldots, l_1, l_2, l_2, \ldots. l_2, \ldots\ldots\ldots\ldots, l_k, l_k \ldots\ldots\ldots., l_k \}$ di N_2 termini costituita da n_1 lunghezze l_1, n_2 lunghezze l_2 e così via.
Su questa sequenza operiamo come segue:

1. predisponiamo N_2 verghe V_1 V_2 … V_{N2}, alle quali inizialmente non è associato nessuno dei pezzi richiesti
2. prendiamo in esame il primo termine della sequenza S e associamolo alla prima verga.
3. prendiamo in esame il termine successivo della sequenza S e, esaminando le verghe in successione a partire dalla prima, associamolo alla prima verga in grado di contenerlo (la somma dei termini attribuiti a una verga non deve superare L).
4. ripetiamo il passo 3 sino a esaurire tutti i termini di S.
5. contiamo le verghe alle quali è associato almeno un pezzo, e sia N il risultato.

N costituisce una soluzione del problema, che è certamente quella ottimale se N = N1. Se N > N1, la soluzione ottimale è comunque compresa tra N1 e N.

Riportiamo una semplice applicazione "manuale" del procedimento:

Numero pezzi e relative lunghezze

2	47
1	34
4	32
1	9
7	6

L	100
Somma lunghezze richieste	307
N_1	4
N_2	15

S = { 47, 47, 34, 32, 32, 32, 32, 9, 6, 6, 6, 6, 6, 6, 6 }

Assegnazione dei pezzi alle varie verghe

V_1	$\Sigma = 100$	47 47 6
V_2	$\Sigma = 98$	34 32 32
V_3	$\Sigma = 97$	32 32 9 6 6 6 6
V_4	$\Sigma = 12$	6 6

N = 4

Il primo termine di S { }, 47, viene assegnato alla prima verga, nella quale resta disponibile uno spazio pari a 53. Il secondo termine viene ancora assegnato alla prima verga, nella quale resta disponibile uno spazio pari a 6. Il terzo termine, 34, supera lo spazio disponibile nella prima verga, e viene quindi assegnato alla seconda … e così via.

In questo caso $N = N_1$, per cui N = 4 costituisce la soluzione ottimale.

Il procedimento può essere migliorato in base alla considerazione che il risultato dipende dall'ordine in cui sono disposti i termini di S { }.

Applicando quindi ripetutamente il procedimento dopo aver ogni volta permutato in modo casuale i termini di S { } è possibile in molti casi abbassare il valore di N, migliorando il primo risultato ottenuto.

L'idea di esplorare tutte le permutazioni dei termini di S { } non è praticabile poiché il loro numero cresce molto rapidamente con il numero dei pezzi richiesti. Tuttavia, come mostreremo, un numero ragionevole di iterazioni consente già notevoli miglioramenti.

La procedura descritta è stata implementata in un programma per PC.

Il programma è stato eseguito su 1000 richieste, generate casualmente, delle seguenti caratteristiche:

- lunghezza della verga standard **100**
- lunghezze richieste **6**, scelte casualmente entro i limiti di **10** e **50** (il limite superiore di 50 è stato ritenuto il massimo ragionevolmente compatibile con lunghezza della verga standard = 100)
- pezzi per ciascuna lunghezza scelti casualmente entro i limiti da **1** e **10**

Le 1000 richieste sono state elaborate effettuando un numero di iterazioni pari a 1, 10, 100, 1000 e 10000.

La tabella seguente mostra come si è distribuito il valore N - N1 dopo le elaborazioni.

applicazioni	1	10	100	1.000	10.000
N-N1					
0	55	178	316	412	500
1	374	517	530	491	433
2	404	257	130	84	55
3	147	45	20	12	9
4	19	3	3	1	3
5	1	0	1	0	0
totali	1.000	1.000	1.000	1.000	1.000

E' evidente il miglioramento dei risultati dovuto all'applicazione ripetuta dell'algoritmo. Il numero dei risultati certamente ottimali, cioè quelli per N = N1, passa da 55 nel caso di una sola applicazione per crescere macroscopicamente sino a raggiungere 500 nel caso di 10000 applicazioni. Con l'aumentare delle applicazioni si restringono inoltre i limiti entro i quali può variare il valore ottimale di N. Nel caso di 1000 iterazioni per ben 412 + 491 = 903 casi il valore di ottimale di N è compreso tra N_1 e N_1+1

Il problema posto riveste notevole interesse nell'industria. Sono stati sviluppati numerosi software applicativi che forniscono il piano di taglio ottimizzato non solo per oggetti lineari (verghe, tubi), ma anche per oggetti a due e tre dimensioni.
Per il caso lineare, alcuni di questi software sono reperibili effettuando, ad esempio su Google, una ricerca con le parole chiave " **1D cutting optimization**".