"Sapienza" Università di Roma - Dipartimento di Ingegneria Informatica, Automatica e Gestionale

Introduzione all'Ottimizzazione Combinatoria

Docente: Renato Bruni

bruni@dis.uniroma1.it

Corso di: Ottimizzazione Combinatoria

Problemi di Decisione: Alternative

I problemi di decisione sono molto diffusi e possono essere di molte diverse tipologie, però hanno tutti qualcosa in comune:

occorre scegliere tra un insieme di alternative, di norma ben distinte e non mescolabili …

$$S = \textit{insieme delle possibili alternative}$$

Esempi:

Scelta del **modo di trasportare qualcosa**:

$S = \{\textit{Nave + Treno, Camion, Aereo+Treno}\}$

Scelta del **periodo di produzione**:

$S = \{\textit{Gennaio, Febbraio+Marzo, Marzo+Aprile, Luglio+Agosto}\}$

Scelta del **percorso più breve da** s **a** t :

$S = \{\textit{percorsi da } s \textit{ a } t\}$

Scelta della **ditta a cui affidare un lavoro** :

$S = \{\textit{ditte disponibili}\}$

e Funzione Obiettivo

… e abbiamo un criterio quantitativo per scegliere

$f(x) : S \rightarrow \boldsymbol{R}$ = *funzione obiettivo*

normalmente la funzione più semplice che rappresenti bene il nostro criterio

Esempio:

Scelta **in base al costo**

$S = \{Nave + Treno,\ Camion,\ Aereo+Treno\}$

$f(Nave+Treno) = 150.000\ €$
$f(Camion) = 121.000\ €$
$f(Aereo+Treno) = 180.000\ €$

Modello di Ottimizzazione

Problema di Decisione = $\min \{ f(x) : x \in S \}$

Variabili di decisione: x (*rappresenta ciò che devo decidere*)

Soluzione ottima: $x^*: f(x^*) \leq f(x)$ per ogni $x \in S$

Soluzione ammissibile: ogni $x \in S$

$$f(x^*) = \min \{ f(x) : x \in S \}$$

$$x^* = \text{argmin} \{ f(x) : x \in S \}$$

Esempio: Scelta del modo di trasportare qualcosa:

$S = \{Nave+Treno, Camion, Aereo+Treno\}$

$f(Nave+Treno) = 150.000$ €

$f(Camion) = 121.000$ €

$f(Aereo+Treno) = 180.000$ €

f(x*)=121.000

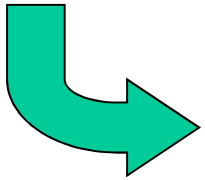
x* = Camion

Assumiamo Numero Finito di Soluzioni

Problema di Decisione = $min\ \{ f(x) : x \in S \}$

con $(1<)\ |S| < \infty$

Cioè l'insieme delle soluzioni ammissibili ***ha cardinalità finita***



L'**ottimo** x^* esiste e può essere individuato con una **enumerazione completa** di S

$|S| < \infty$ è una *ipotesi restrittiva* ?

Le componenti di ogni aspetto di un problema di decisione possono assumere un insieme finito di valori e dunque abbiamo sempre un insieme finito di soluzioni

Allora è facile risolvere?

- Vediamo un esempio dovuto a **George Dantzig** (1914-2005), uno di fondatori della Ricerca Operativa, in particolare della programmazione lineare, e inventore del metodo del simplesso

Abbiamo una azienda con 70 persone e 70 lavori da fare

- Vogliamo **assegnare** i lavori alle persone in modo che ognuno ne faccia uno, ed abbiamo una valutazione $c(p, l)$ di quanto è conveniente ogni possibile assegnamento persona p a lavoro l
- Vogliamo la soluzione che ed es. massimizzi la somma delle convenienze degli assegnamenti fatti, cioè scegliere una soluzione tra tutte le **permutazioni** di 70 elementi
- Sono $70! \sim 10^{100}$

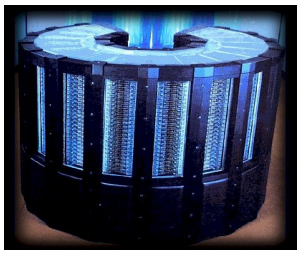
Se proviamo a valutare tutte le 70! possiamo risolvere?

- In teoria il ragionamento sembra funzionare. Ma in **pratica**? Vediamo cosa ha stimato Dantzig riguardo alla soluzione di questo problema per enumerazione completa

Esempio 70x70

- Proviamo con un **computer** capace di lavorare a 1MHz (ora obsoleto, all'epoca dell'esempio futuristico)

- Dopo diversi giorni di calcolo non ha finito e lo stoppiamo…
- Allora immaginiamo di dargli **più tempo**: diciamo che lo abbiamo fatto partire al momento della nascita dell'universo (big bang). Ad oggi avrebbe finito? La risposta è no!
- Allora prendiamo un computer **più veloce**: un supercomputer a 1Thz (forse ancora non esiste, ma prima o poi…)



- Immaginiamo di averlo fatto partire al momento del big bang. Ad oggi avrebbe finito? La risposta è ancora no!
- Allora di computer ne prendiamo tanti che lavorano **in parallelo**: tappezziamo le terra di computer di questo tipo che lavorano dal momento del big bang. Ad oggi avrebbe finito? La risposta è ancora no!

Servono 10^{40} terre così equipaggiate per risolvere enumerando!

Come Risolvere?

- Eppure problemi con 70 persone e 70 lavori si risolvono facilmente
- Come? Servono **i modelli e gli algoritmi giusti** …
- La scelta del modello e dell'algoritmo spesso determina enormi differenze nell'uso delle risorse di calcolo, e quindi nei tempi di soluzione
- Modelli e algoritmi che pure sono teoricamente corretti possono essere estremamente inefficienti in pratica …
- L'oggetto di questo corso saranno proprio i modelli e gli algoritmi da usare per **diverse categorie di problemi molto importanti** in campo applicativo: i problemi di **Ottimizzazione Combinatoria**

- What is an algorithm? Algorithms are what make it possible for computers to resolve problems. Algorithms are at the very heart of computer science.
- An informal definition (from WikiPedia):

  "An **algorithm** is a procedure (a finite set of well-defined instructions) for accomplishing some task which, given an initial state, will terminate in a defined end-state"

Campi di Applicazione dell'OC

Moltissimi **problemi di decisione del mondo reale** possono essere formulati come problemi di **Ottimizzazione Combinatoria** (branca della Ricerca Operativa)…

… perché questa disciplina è costituita di un insieme di metodi, che possono essere applicati a problemi delle più diverse aree

- Biologia molecolare
- Selezione di investimenti
- Network design
- Localizzazione di impianti
- Pianificazione della produzione
- Instradamento di veicoli
- Data mining
- Ricostruzione di informazione
- Deduzione automatica
- Assegnazione di frequenze
- Sequenziamento di lavori
- etc. etc.

Come in tutta la Ricerca Operativa, si lavora usando l'**approccio modellistico:** si fa un modello (matematico) del problema e poi si usano algoritmi già studiati per quei tipi di modelli, anziché trovare un algoritmo nuovo per ogni nuovo problema

Problemi di OC con FO lineare

Un problema di **Ottimizzazione Combinatoria** ha questa **struttura matematica**:

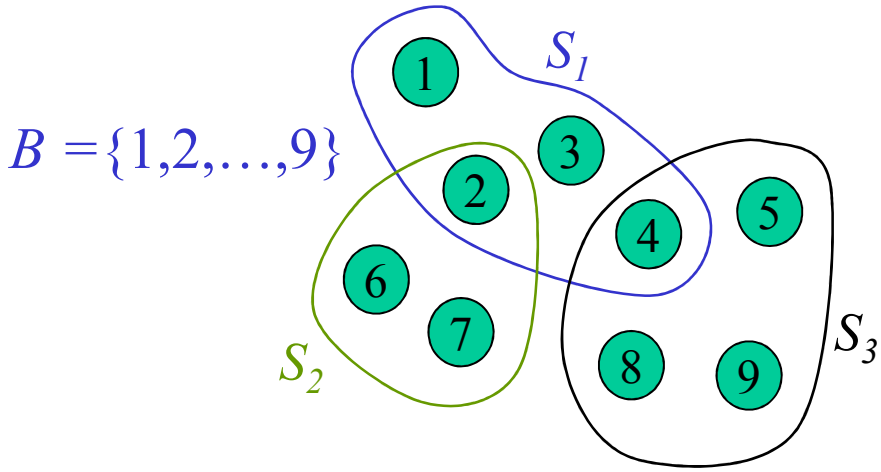
- Abbiamo un insieme **base**: $B=\{b_1, \ldots, b_n\}$
  (sono gli eventi elementari, per es: progetto i attivato, connessione j stabilita)
- Ogni **soluzione ammissibile** è un sottoinsieme $S \subseteq B$ (insieme di eventi elementari che rispettano le condizioni date)
- L'insieme delle soluzioni ammissibili è $\mathcal{S}=\{S_1, \ldots, S_m\} \subseteq \mathcal{P}(B)$ (card 2^n)
- Ogni elemento di B ha il suo costo (o vantaggio) elementare $\{c_i\}$
- Abbiamo una funzione di **costo** $c: S \rightarrow \mathrm{R}$,

(nel nostro caso lineare: la somma dei c_i dei componenti b_i della soluz. ammiss.)

$$c(S_1)=\sum_{i \in S_1} c_i$$

Vogliamo trovare:

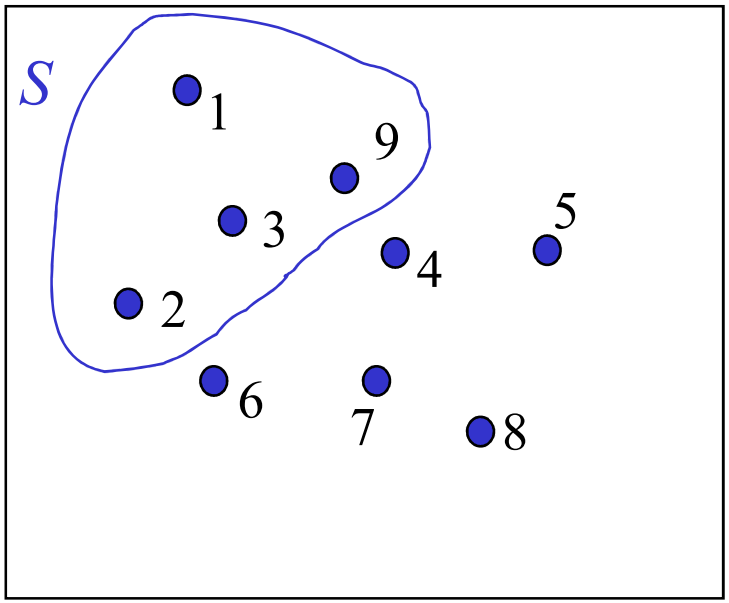
$$min\ \{c(\mathrm{S}):\ S \in \mathcal{S}\}$$



Corrispondenza coi Problemi di PL01

- Ad esempio, consideriamo un insieme base $B = \{1,2,\ldots,9\}$
- e una soluz ammissibile $S = \{1,2,3,9\} \in \mathcal{S}$
- Se **rappresentiamo** S con il suo vettore di incidenza x^S abbiamo

$$x^S = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$



E quindi c'è corrispondenza tra

$\{sottoinsiemi\} = \mathcal{S}$ e $\{vettori\ di\ incidenza\ dei\ sottoinsiemi\} = V$

Problemi di OC con funz obiettivo lineare e PL01 $c(S)$ e $c^T x$

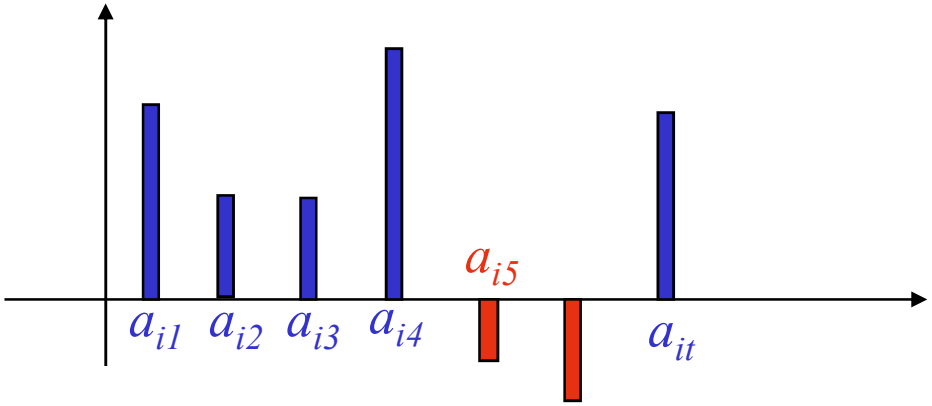
$min\ \{c(S):\ S \in \mathcal{S}\}$ e $min\ \{c^T x : x \in V \subseteq \{0,1\}^n\}$

Ogni problema di OC con FO lineare può essere scritto come PL01

Esempio: Pianificazione di investimenti

Abbiamo:

- Insieme $I = \{1, 2, \ldots, n\}$ di ***Investimenti***
- ***Indice di redditività*** (vantaggio) c_i dell'investimento i
- La redditività totale di un insieme di investimenti F è $\sum_{i \in F} c_i$
- Orizzonte temporale $T = \{1,2,\ldots,t\}$
- ***Vettore dei flussi di cassa*** $a_i = (a_{i1}, a_{i2}, \ldots, a_{it})$ dell'investimento i
- ***Budget*** b_j del periodo $j \in T$ (quanto possiamo spendere in quel periodo)



Il Problema di Decisione

- Esempio di **flussi di cassa** di un problema con 3 investimenti

1: 10, 10, -12, 10, -20

2: -3, 5, 10, -20, 10, -20

3: 5, 5, 5, 0, -5, -10, -20

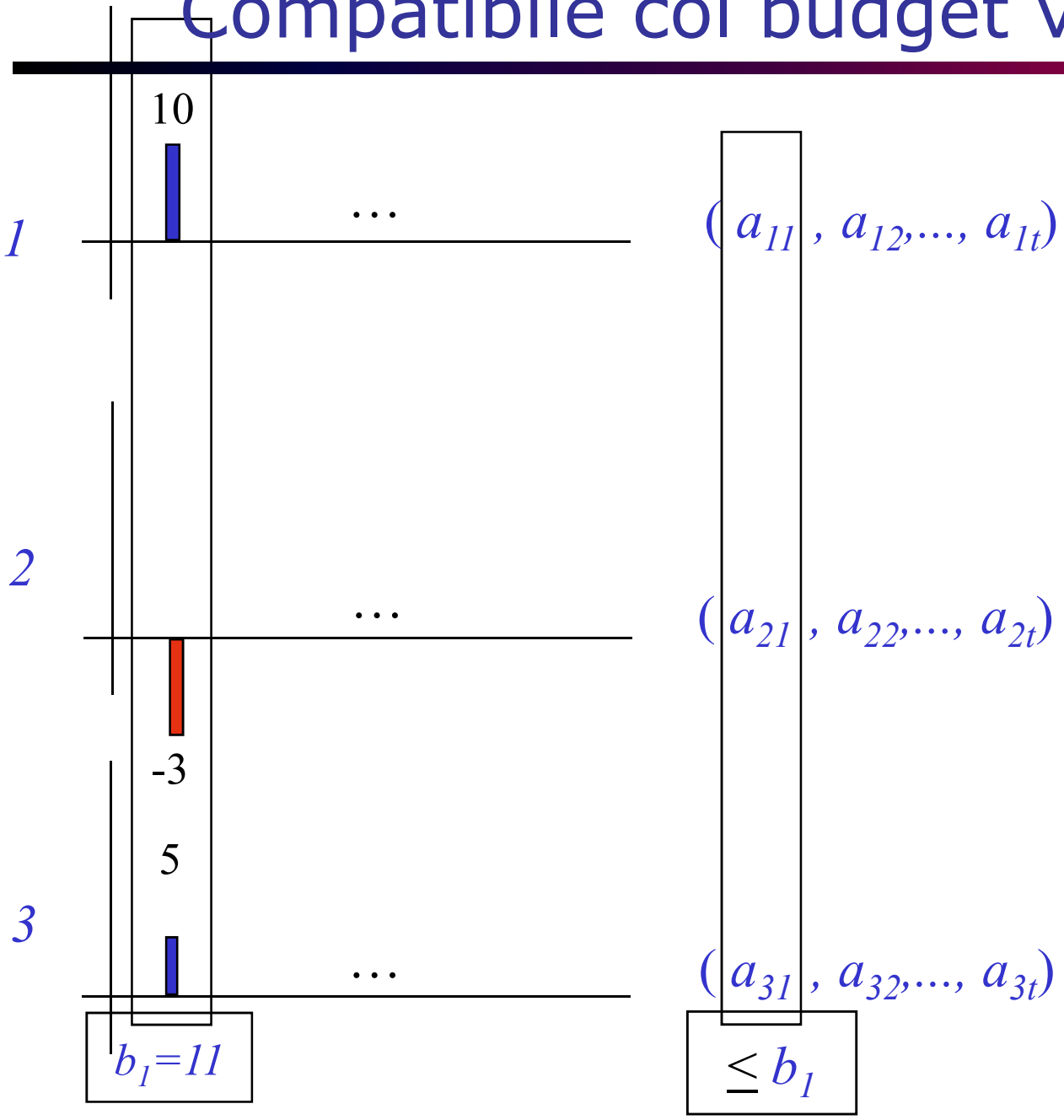
- Esempio di **budget** su 7 periodi $b=\{11, 20, 20, 20, 20, 0, 0\}$

TROVARE

Insieme di investimenti F^* di redditività totale massima e tale che la somma dei flussi di cassa a_{ij} dei progetti attivati sia, in ogni periodo $j \in T$, compatibile col budget b_j

S = {*Vettori di incidenza degli insiemi di progetti compatibili con i vincoli di "budget" in ogni periodo*}

Compatibile col budget vuol dire …



quindi
nell'esempio
si ha che:

$$\begin{bmatrix}1\\0\\1\end{bmatrix} \notin S$$

$$\begin{bmatrix}1\\1\\0\end{bmatrix} \in S$$

Struttura del problema

- Ogni investimento corrisponde a un elemento dell'**insieme base** (*nell'esempio B ha 3 elementi*)
- Ogni insieme di investimenti ammissibili è un sottoinsieme dell'insieme di tutti gli investimenti, quindi corrisponde a **un sottoinsieme dell'insieme base** ($S \subseteq B$: *per esempio, se è ammissibile fare insieme gli investimenti 1 e 2, si ha una soluzione ammissibile* $\{1,2\} \subseteq \{1,2,3\}$)
- Vogliamo massimizzare la **funzione lineare** somma delle redditività degli investimenti attivati ($c(S) = \sum_{i \in Attivati} c_i$)

- Quindi questo problema di decisione è un problema di **OC con FO lineare**
- Inoltre, ogni insieme di investimenti ammissibili (=soluzione ammissibile) è rappresentabile con un **vettore binario**
- Quindi è anche un problema di **PL01** (vedi modello seguente)

Modello dell'Esempio

Esempio: ***flussi di cassa*** a_{ij}

1: 10, 10, -12, 10, -20 , 0, 0

2: -3, 5, 3, -20, 10, -20 , 0

3: 5, 5, 5, 0, -5, -10, -20

Ci sono al massimo 7 periodi

Mettiamo degli 0 dove non ci sono flussi di cassa (anche per confrontare progetti con diversa periodicità)

- ***Budget*** dei vari periodi $b=\{11, 20, 20, 20, 20, 0, 0\}$
- Orizzonte temporale $T=\{1,...,7\}$
- Variabili di decisione $x = \{x_1, x_2, x_3\}$

Modello di PL01
con 3 var e 7 vinc

$$
\begin{cases}
max\ c_1 x_1 + c_2 x_2 + c_3 x_3 \\
10 x_1 - 3 x_2 + 5 x_3 \leq 11 \\
10 x_1 + 5 x_2 + 5 x_3 \leq 20 \\
-12 x_1 + 3 x_2 + 5 x_3 \leq 20 \\
10 x_1 - 20 x_2 + 0 x_3 \leq 20 \\
-20 x_1 + 10 x_2 - 5 x_3 \leq 20 \\
0 x_1 - 20 x_2 - 10 x_3 \leq 0 \\
0 x_1 + 0 x_2 - 20 x_3 \leq 0 \\
x_i \in \{0,1\},\ i = 1,2,3
\end{cases}
$$

Possibilità offerte dal Modello

- *Indice di redditività* c_i di ogni progetto è calcolabile offline anche con metodi sofisticati (vedi economisti)
- L'aggiunta di altri vincoli non invalida il modello
- Ad esempio, esprimiamo ogni tipo di relazione logica tra i progetti aggiungendo al modello visto equazioni o disequazioni lineari (elevato potere espressivo)

Esempi:

$x_i \leq [\geq, =] \; x_j$ ⇒ • i può essere svolto *solo se* [è svolto *se,* è svolto *se e solo se*] j viene svolto

$x_i + x_j \leq 1$ ⇒ • *solo uno tra* i e j può essere svolto (ad es. se i e j sono copie dello stesso progetto con stessi flussi ma diversi inizi)

$x_i = (1 - x_j)$ ⇒ • i viene svolto *se e solo se non* viene svolto j

$x_i \leq x_h + x_k + (1 - x_j)$ ⇒ • i può essere svolto *solo se* h *o* k vengono svolti *o* j *non* viene svolto (almeno una delle 3 condiz.)

$x_i \leq (x_h + x_k + (1 - x_j))/3$ ⇒ • i può essere svolto *solo se* h *e* k vengono svolti *e* j *non* viene svolto (tutte e 3)

Richiami di Calcolo Combinatorico

Studia i modi per raggruppare e/o ordinare gli elementi di un insieme finito di oggetti – ci servirà

Permutazioni di n oggetti: modi di ordinare n oggetti, tutti diversi

Per la prima posizione posso scegliere n oggetti, per la seconda n-1 e così via: sono $n(n-1)\ldots1 = n!$

Se un oggetto è presente k volte, e non voglio contare le permutazioni in cui cambia solo l'ordine di tale oggetto, devo dividere per k! : sono n!/k!

Disposizioni di n oggetti di classe k: modi di ordinare n oggetti in k posizioni, senza ripeterli

Per la prima posizione posso scegliere n oggetti, per la seconda n-1 e così via: sono $n(n-1)\ldots(n-k+1) = n! / (n-k)!$

Disposizioni di n oggetti di classe k con ripetizioni: modi di ordinare n oggetti in k posizioni, anche ripetendoli

Come le cifre di un numero: sono n^k

Combinazioni di n oggetti di classe k: modi di raggruppare (l'ordine non conta) n oggetti in k posizioni, senza ripeterli

Sono le Disp(n,k) / Perm(k) = n! / (n-k)! k! = $\binom{n}{k}$

Combinazioni di n oggetti di classe k con ripetizioni: modi di raggruppare n oggetti in k posizioni, anche ripetendoli

Sono le Disp(n+(k-1) [*dato che ripeto è come se avessi più oggetti*], k) / Perm(k) = $\binom{n+k-1}{k}$