1 Problema

Il problema nasce dalla necessità di una strategia veloce e corretta per la creazione di una scheda di palestra che rispetti determinati vincoli fisici. L'obiettivo è quello di fornire un algoritmo che riesca a rielaboare velocemente una scheda in seguito alla modifica di uno qualsiasi dei parametri di essa. I vincoli da rispettare sono:

- Il numero di giorni di allenamento alla settimana, adattato all'esigenza dell'utente
- Per ogni giorno, il numero massimo di serie, oltre il quale l'energia posseduta dal corpo non è più sufficiente per una esecuzione efficace di un qualsiasi esercizio
- Per ogni muscolo, il numero massimo di serie giornaliere, oltre il quale il muscolo non è più in grado di rispondere in maniera ottimale
- Per ogni muscolo, il numero minimo di serie settimanali, senza le quali il muscolo non cresce in maniera ottimale

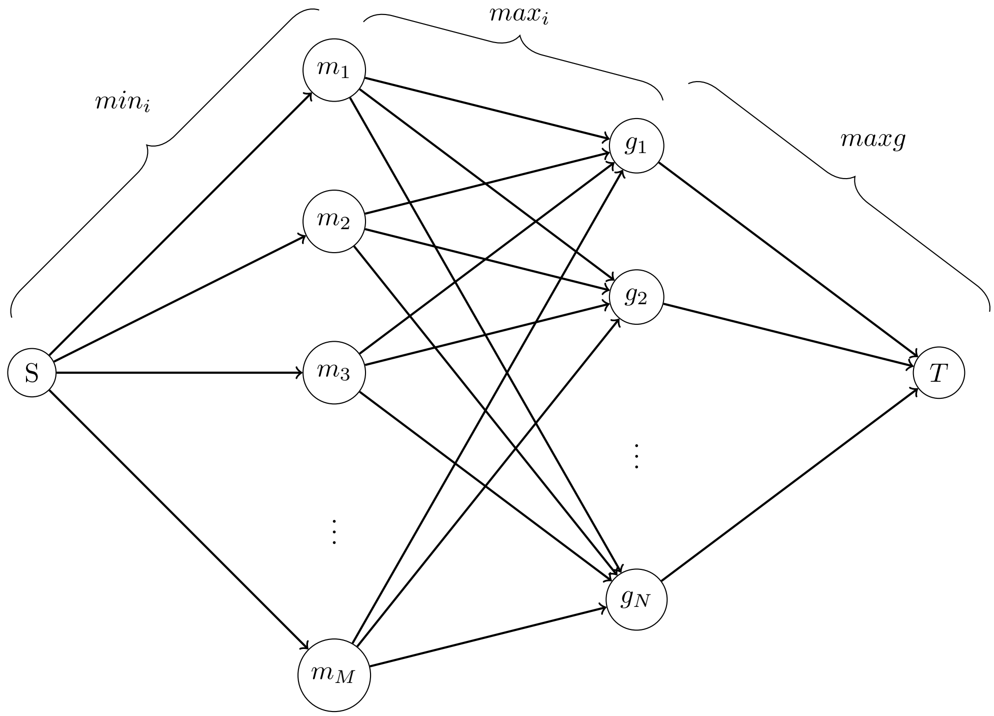
2 Formalizzazione

Dati N giorni $g_1, g_2, ..., g_N$, dove ogni giorno ha un massimo di serie giornaliere pari a $maxg$ e M muscoli $m_1, m_2, ..., m_M$, dove ogni muscolo ha un massimo di serie giornaliere $max_1, max_2, ..., max_M$ e un minimo di serie settimanali $min_1, min_2, ..., min_M$. L'algoritmo deve ritornare un insieme S di assegnazioni giorno, muscolo, serie (g, m, s) tale che valga:

1. $g \leq N$, $m \leq M$
2. $\forall i \in \{1...N\} \sum_{s \in \{S|g=i\}} s \leq maxg$ (per ogni giorno il numero di serie deve essere inferiore al massimo giornaliero)
3. $\forall i \in \{1...M\} \sum_{s \in \{S|m=i\}} s \geq min_i$ (per ogni muscolo il numero di serie settimanali deve essere almeno pari al minimo settimanale)
4. $\forall i \in \{1...N\}, j \in \{1...M\} \sum_{s \in \{S|g=i,m=j\}} s \leq max_i$ (per ogni giorno, ogni muscolo non può superare il suo massimo giornaliero)

3 Implementazione

L'algoritmo fa utilizzo della tecnica di Ford-Fulkerson per la ricerca del flusso massimo. Viene creato un grafo del seguente tipo:



Il costo computazionale per la creazione del grafo è $O(MN)$. È possibile automatizzare la scelta di $maxg$ in modo da uniformare il numero di serie giornaliere, questo viene fatto assegnando a $maxg$ il valore atteso del numero di serie giornaliere $\lceil \sum_{i=1}^{M} min_i/N \rceil$ con costo $O(M)$. Successivamente viene applicato l'algoritmo di Ford-Fulkerson.

- Il vincolo 1 è garantito dalla definizione del problema
- Per la verifica del vincolo 2 gli archi (g_j, T) $\forall j \in 1...N$ hanno capacità pari a $maxg$
- Il vincolo 3 invece deve essere verificato manualmente, controllando che valga $w(S, m_i) \geq min_i$ $\forall i \in 0...M$ in tempo $O(M)$.
- Il vincolo 4 è verificato dal fatto che gli archi (m_i, g_j) $\forall i \in 0...M, j \in 0...N$ hanno capacità pari a max_i

Infine vengono presi gli archi di tipo (m_i, g_j) con flusso positivo, che corrisponderanno alle assegnazioni. Le assegnazioni vengono poi ordinate per giorno in tempo $O((NM) \log NM)$. Ricapitolando gli step dell'algoritmo sono:

- Creazione del grafo $O(MN)$
- Calcolo di $maxg$ $O(M)$

- Applicazione algoritmo di Ford-Fulkerson $O((M+N)^2|f^*|)$
- Verifica di $w(S, m_i) \geq min_i \; \forall i \in 0...M \; O(M)$
- Ricerca assegnazioni $O(MN)$
- Ordinamento assegnazioni $O((MN)\log(MN))$

L'algoritmo finale ha costo complessivo pari alla somma dei precedenti:

$$
\begin{aligned}
T(M,N) &= O((MN) + (M) + ((M+N)^2|f^*|) + (M) + (MN) + (MN)\log(MN)) \\
&= \boxed{O((M+N)^2|f^*| + (MN)\log(MN))}
\end{aligned}
$$

4 Post-processing

Al termine dell'algoritmo ci troviamo con una collezione di record (giorno, muscolo, serie), ognuno di questi record può essere suddiviso in più esercizi. Dati i valori K numero di serie del record e $mins$ e $maxs$ rispettivamente il numero minimo e massimo di serie per esercizio, dobbiamo trovare un insieme S tale che:

- $\sum_{s \in S} s = K$
- $mins \leq s_i \leq maxs \; \forall s \in S$
- $|S|$ sia il più piccolo posssibile
- Il valore di $d = \sum_{s_1, s_2 \in S} |s_1 - s_2|$ sia il più piccolo possibile (la differenza tra il numero di serie tra due esercizi sia minimizzata)

Per fare ciò possiamo applicare il seguente algoritmo greedy:

- Poniamo $|S| = \lceil \frac{K}{maxs} \rceil$ con ogni elemento di S pari a $maxs$
- Prendiamo un $s \in S$ diminuiamo s di uno fino a quando $\sum_{s \in S} s$ diventa K oppure fino a quando s diventa $mins$
- Se alla fine dell'algoritmo $\sum_{s \in S} \neq K$ non esiste una soluzione e ritorniamo $\{K\}$
- L'ultimo passo è quello di minimizzare la differenza tra i vari $s \in S$ per fare ciò poniamo $s_1 = \lfloor (s_1 + s_1)/2 \rfloor$ e $s_2 = \lceil (s_1 + s_1)/2 \rceil \; \forall s_1, s_2 \in S$.

L'algoritmo ha complessità $O(MN * (maxs - mins) * \lceil K/maxs \rceil + \lceil K/maxs \rceil^2)$. Supponendo che $mins$ e $maxs$ non facciano parte dell'input la complessità diventa $O(MNK + K^2)$.

5 Dimostrazione proprietà di scelta greedy (TODO)