
UNIVERSITÀ DEGLI STUDI DI CATANIA

DIPARTIMENTO DI MATEMATICA E INFORMATICA

CORSO DI LAUREA MAGISTRALE IN INFORMATICA

---

# UNCAPACITATED FACILITY LOCATION PROBLEM WITH IMMUNOLOGICAL ALGORITHM

---

PROGETTO DI COMPUTAZIONE NATURALE

---

Andrea Sequenzia
Matricola: W82000160

---

ANNO ACCADEMICO 2018/2019

# Indice

# 1 Introduzione

# 2 Definizione del problema

In un problema UFLP, ci sono $m$ clienti da servire ed $n$ possibili location dove poter stabilire delle facilities che servano i clienti e soddisfare la domanda. Ogni facility $i$ ha un costo fisso di installazione $f_i$. Un costo di trasporto $c_{ij}$ per servire il il cliente $j$ dalla facility $i$. Le facility non hanno limiti nelle capacità di quanto possono servire e non ci devono essere clienti non soddisfatti. L'obiettivo finale è quello di stabilire quali facility attivare, soddisfando tutti i clienti, minimizzando il costo per ottenere ciò. Formalmente:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} + \sum_{i=1}^{n} f_i y_i \\
\text{subject to} \quad & \sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, ..., m, \\
& x_{ij} \leq y_i, \quad i = 1, ..., n, j = 1, ..., m, \\
& x_{ij}, y_i = 0, 1, \quad i = 1, ..., n, j = 1, ..., m
\end{aligned}
$$

dove

$c_{ij} =$ il costo per soddisfare la domanda del cliente $j$ dalla facility $i$;

$f_i =$ il costo per stabilire la facility nella location $i$

$$
x_{ij} = \begin{cases} 1 & \text{se il cliente j viene servito dalla facility i,} \\ 0 & altrimenti; \end{cases}
$$

$$
y_i = \begin{cases} 1 & \text{se la facility è stabilita nella location i,} \\ 0 & altrimenti; \end{cases}
$$

Il problema può essere matematicamente decomposto in due sottoproblemi indipendenti [1]:

1. **Location**, ovvero le facilities da stabile ($y_i$)

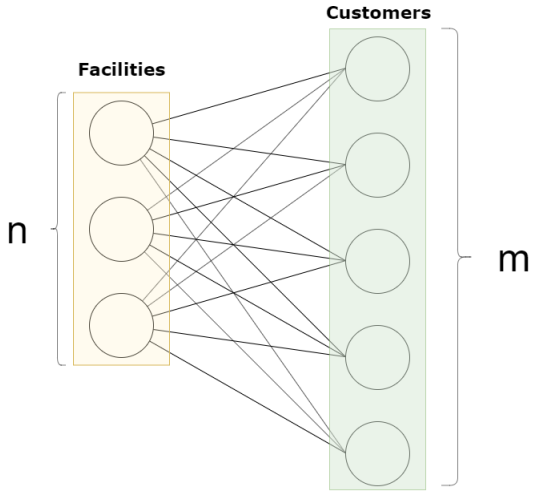


Figura 1: Esempio problema UFLP

2. **Allocation**, per le facility stabilite, determinare la distribuzione e quindi quali archi usare per ogni customer ($x_{ij}$)

Per ogni soluzione del problema di Location, una soluzione ottima può essere ottenuta per il problema Allocation. Dato un vettore $y$, può essere ottenuto un assegnamento ottimale delle $x_{ij}$ usando la seguente formula:

$$k : \min_{k} c_{kj}, \quad k = 1, ..., n$$

$$x_{ij} = \begin{cases} 1 & \text{se i=k,} \\ 0 & altrimenti; \end{cases}$$

Quindi il problema si riduce a determinare l'assegnamento di facilities ottimale.

# 3 Algoritmo immunologico

Un algoritmo immunologico è un algoritmo basato sul clonal selection principle, esso è un ottimo esempio di strategia intelligente in cui l'adattamento opera a livello locale, mentre un

comportamento complesso agisce a livello globale. L'idea di questa tipologia di algoritmo deriva dal sistema immunitario naturale, in particolare da come le cellule si adattano per legarsi ed eliminare entità straniere, meglio conosciute come Antigene. Questo algoritmo è basato su due entità: gli **antigeni** che rappresentano il problema da affrontare e le **B cell** (Linfocita B) che rappresentano un punto nello spazio di ricerca [2]. L'algoritmo crea una popolazione di B cell, esse cresceranno e si perfezioneranno nel corso di un numero prefissato di generazioni, morendo quando hanno raggiunto un'età prefissata. La qualità delle B cell si valuta attraverso una funzione di fitness. Gli operatori usati da questo algoritmo sono in sequenza i seguenti:

1. **Cloning**: questo operatore simula il meccanismo di proliferazione di un sistema immunitario. Esso genera una popolazione intermedia $P^{(clo)}$ copiando *dup* volte ogni B cell, assegnando un'età, la nuova popolazione così fatta sarà delle dimensioni $d \times dup$.

2. **Inversely Hypermutation**: il compito di questo operatore è quello di esplorare il vicinato di ogni clone creato dal precedente operatore. Ciò viene compiuto eseguendo $M$ mutazioni ad ogni elemento di $P^{(clo)}$. Per determinare $M$, viene adottata una legge inversamente proporzionale al valore della fitness: più piccolo (minimizzazione) è il valore valore della funzione di fitness, meno mutazioni saranno fatte alla B cell. Dato un clone $x$, per determinare il valore $M$ è necessaria la seguente formula:

$$\alpha = e^{-\rho \hat{f}(x)} \tag{1}$$

   dove $\alpha$ rappresenta il mutation rate,$\rho$ lo shape del mutation rate e $\hat{f}(x)$ è il valore della funzione di fitness normalizzato tra $[0, 1]$. Il numero di mutazioni $M$ è dato da:

$$M = \lfloor (\alpha \times l) + 1 \rfloor \tag{2}$$

   dove $l$ è la lunghezza di una B cell.

3. **Aging**: questo operatore aiuta l'algoritmo a saltare via dagli ottimi locali. Per ottenere ciò, rimuove le vecchie B cell dalla popolazione iniziale e da quella ipermutata. L'eliminazione avviene quando l'età (che viene incrementata ad ogni fine generazione) raggiunge

l'età massima ($\tau_b$). Così facendo, si produce alta diversità e si aiuta l'algoritmo ad evitare le convergenze premature. Viene anche fatta un eccezione nella rimozione, ovvero lasciare la migliore soluzione nella popolazione anche se ha raggiunto la massima età. Questa variante viene chiamata *elitist aging operator*.

4. ($\mu + \lambda$)**-Selection**:

# 4 Implementazione

# 5 Esperimenti

# 6 Conclusioni

# Riferimenti bibliografici

[1] K.S. Al-Sultan and M.A. Al-Fawzan. A tabu search approach to the uncapacitated facility location problem. *Annals of Operations Research*, 86(0):91–103, Jan 1999.

[2] Mario Pavone and Giuseppe Nicosia. Clonal selection - an immunological algorithm for global optimization over continuous spaces. *Journal of Global Optimization*, 53:1–40, 08 2011.