

Uncapacitated Facility Location Problem

PROGETTO

Laboratorio di Intelligenza Artificiale

Carolina Crespi

3 Settembre 2026

1 Uncapacitated Facility Location Problem – UFL

Il *Uncapacitated Facility Location Problem* (UFL) è un classico problema di ottimizzazione combinatoria, classificato come NP-hard, con numerose applicazioni in logistica, progettazione di reti e pianificazione territoriale.

Intuitivamente, il problema modella la situazione in cui, dato un insieme di siti candidati per l'apertura di *facility* (magazzini, negozi, server, antenne) e un insieme di *clienti* da servire, si vuole scegliere quali facility aprire per minimizzare la somma di due contributi: il costo fisso di apertura delle facility scelte e il costo variabile di servizio dei clienti (ciascuno assegnato a una facility aperta). Nella variante *uncapacitated* qui considerata, non ci sono limiti alla capacità di servizio di ciascuna facility. Alcuni esempi applicativi:

- la scelta di dove aprire i punti vendita di una catena commerciale, bilanciando i costi di affitto e la distanza dai clienti potenziali;
- la localizzazione di data center di un cloud provider, considerando costi fissi di installazione e costi di latenza verso gli utenti;
- il posizionamento di stazioni di ricarica per veicoli elettrici, con costi di installazione e costi di accessibilità per l'utenza;
- l'apertura di uffici distaccati di un servizio pubblico, con costi di gestione e costi di spostamento per i cittadini.

Formalmente, siano dati:

- un insieme di m facility candidate, ciascuna con un costo di apertura $f_i > 0$, per $i \in \{1, \dots, m\}$;
- un insieme di n clienti;
- una matrice di costi di servizio $c_{ij} > 0$, che rappresenta il costo di servire il cliente j dalla facility i .

Introducendo le variabili binarie $x_i \in \{0, 1\}$ (con $x_i = 1$ se la facility i è aperta) e $y_{ij} \in \{0, 1\}$ (con $y_{ij} = 1$ se il cliente j è servito dalla facility i), il problema si formula come:

$$\min \sum_{i=1}^m f_i x_i + \sum_{i=1}^m \sum_{j=1}^n c_{ij} y_{ij} \quad (1)$$

soggetta ai vincoli:

$$\sum_{i=1}^m y_{ij} = 1 \quad \forall j \in \{1, \dots, n\} \quad (\text{ogni cliente assegnato a una facility}) \quad (2)$$

$$y_{ij} \leq x_i \quad \forall i, j \quad (\text{un cliente può essere servito solo da una facility aperta}) \quad (3)$$

Va osservato che, dato un insieme $S \subseteq \{1, \dots, m\}$ di facility aperte, l'assegnamento ottimo dei clienti è determinato univocamente: ciascun cliente j viene assegnato alla facility aperta di costo di servizio minimo, ossia $\arg \min_{i \in S} c_{ij}$. Di conseguenza, il problema si riduce a scegliere il sottoinsieme S di facility da aprire, e il costo totale può essere espresso come:

$$\text{costo}(S) = \sum_{i \in S} f_i + \sum_{j=1}^n \min_{i \in S} c_{ij}.$$

La difficoltà del problema deriva dal compromesso tra due tendenze opposte: aprire poche facility riduce il costo fisso di apertura ma aumenta i costi di servizio (i clienti sono in media più lontani); aprire molte facility riduce i costi di servizio ma incrementa il costo fisso. La ricerca dell'ottimo richiede quindi di esplorare in modo intelligente lo spazio 2^m dei possibili sottoinsiemi di facility.

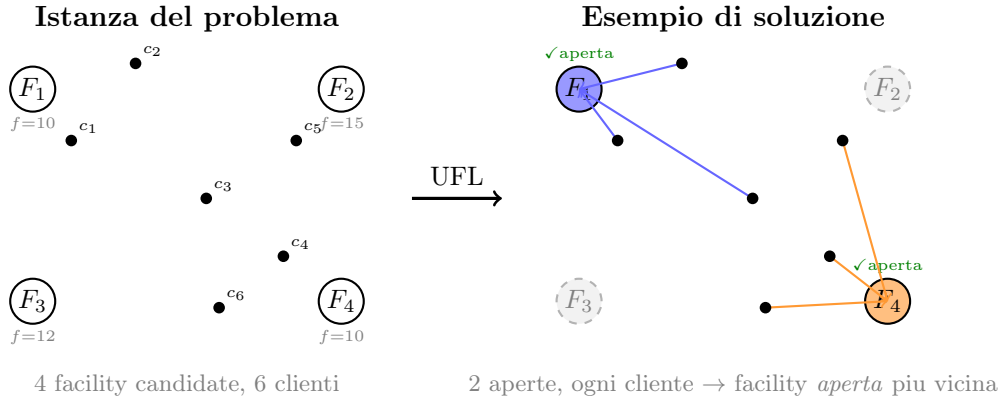


Figura 1: A sinistra: un'istanza illustrativa con 4 facility candidate (cerchi vuoti, con associato il costo di apertura f_i) e 6 clienti (pallini neri). A destra: un esempio di soluzione in cui vengono aperte 2 facility (in colore pieno, con ✓); le facility non aperte sono mostrate in grigio tratteggiato. Ogni cliente è assegnato con una freccia alla facility aperta di costo di servizio minimo. La figura ha scopo unicamente illustrativo e non corrisponde all'istanza considerata nel progetto.

2 Istanza del Problema

Nel progetto è considerata una singola istanza del UFL composta da $m = 6$ facility candidate e $n = 10$ clienti.

I dati dell'istanza sono forniti nel file allegato alla presente traccia:

- **facilities.txt**: contiene il numero di facility m , il numero di clienti n , il vettore dei costi di apertura $f = (f_1, \dots, f_m)$ e la matrice dei costi di servizio c di dimensione $m \times n$, dove c_{ij} è il costo di servire il cliente j dalla facility i .

Il valore ottimo della funzione obiettivo è noto ed è pari a $z^* = 71$. Tale valore può essere utilizzato per verificare la correttezza delle soluzioni ottenute.

3 Protocollo Sperimentale

Il protocollo sperimentale varia in funzione della natura dell'algoritmo scelto.

Algoritmi deterministici (Branch and Bound, Euristica Greedy ADD)

Per gli algoritmi deterministici è richiesta **una sola esecuzione**. Nella relazione finale devono essere riportati:

- il valore della funzione obiettivo ottenuto;
- l'insieme delle facility aperte;
- l'assegnazione di ciascun cliente alla facility che lo serve;
- la scomposizione del costo totale nelle sue due componenti (costi di apertura e costi di servizio).

Per il Branch and Bound devono essere inoltre descritte la strategia di esplorazione adottata e il criterio di *lower bound* utilizzato.

Algoritmi stocastici (Simulated Annealing, Algoritmo Genetico)

Per gli algoritmi stocastici sono richieste **5 run indipendenti**, utilizzando seed diversi. Ogni run è governata dal seguente criterio di arresto:

- il raggiungimento del valore ottimo noto $z^* = 71$, oppure
- il raggiungimento di un numero massimo di **300 valutazioni** della funzione obiettivo.

Una *valutazione della funzione obiettivo* corrisponde al calcolo del costo totale di una soluzione completa e ammissibile (ossia con almeno una facility aperta), comprensivo dei costi di apertura e dei costi di servizio ottimali dati dall'insieme di facility scelto.

Nella relazione finale devono essere riportati:

- il valore della funzione obiettivo ottenuto in ciascuna run;
- l'insieme delle facility aperte e l'assegnazione dei clienti della run migliore;
- la media dei valori finali sulle 5 run;
- un breve commento sul comportamento dell'algoritmo adottato.

4 Algoritmo da Implementare

Il candidato è invitato a scegliere e implementare uno dei seguenti algoritmi:

1. **Branch and Bound (BnB)**, con definizione dell'albero delle decisioni (una variabile x_i per livello, corrispondente all'apertura o chiusura di ciascuna facility) e strategia di esplorazione dello spazio di ricerca (*depth-first*, *best-bound-first* o strategia mista). La strategia e il criterio di *lower bound* adottati devono essere descritti nella relazione.
2. **Euristica Greedy ADD**, in cui la soluzione viene costruita iterativamente a partire da un insieme di facility aperte inizialmente vuoto: a ogni passo si apre la facility la cui inclusione riduce maggiormente il costo totale (considerando congiuntamente il costo di apertura e la riduzione dei costi di servizio dovuta ai clienti che possono ora essere serviti a costo minore); la procedura termina quando nessuna ulteriore apertura riduce il costo. Il criterio di selezione può essere modificato dal candidato (ad esempio, variante *DROP* che parte con tutte le facility aperte e ne rimuove iterativamente una), purché motivato nella relazione.

3. **Simulated Annealing (SA)**, con rappresentazione binaria della soluzione (un bit per facility), vicinato basato sul flip di un bit (apertura/chiusura di una facility), temperatura iniziale, schema di raffreddamento e condizione di equilibrio lasciati alla scelta del candidato. È obbligatorio garantire l'ammissibilità (almeno una facility aperta) di ogni soluzione generata. Tutti i parametri devono essere dichiarati e motivati nella relazione.
4. **Algoritmo Genetico (GA)**, con codifica binaria della soluzione, operatori di selezione, crossover e mutazione lasciati alla scelta del candidato. È obbligatorio garantire l'ammissibilità di ogni soluzione generata, ad esempio mediante procedura di riparazione. Tutti i parametri devono essere dichiarati e motivati nella relazione.

5 Note Finali al Progetto

La relazione finale deve essere redatta in L^AT_EX, con una lunghezza minima di 4 pagine e massima di 12 pagine.

È fortemente sconsigliata l'inclusione di codice implementato, mentre è fortemente consigliato l'uso di pseudo-codice e diagrammi esplicativi.

La relazione deve illustrare in modo chiaro:

- l'algoritmo sviluppato e le sue caratteristiche principali;
- le motivazioni delle scelte progettuali effettuate;
- eventuali elementi di novità e originalità introdotti;
- un'analisi critica dei risultati ottenuti;
- considerazioni finali sul comportamento dell'algoritmo.

La valutazione finale si baserà principalmente su:

- correttezza dell'implementazione e rispetto dei vincoli;
- qualità dei risultati ottenuti;
- qualità espositiva della relazione;
- completezza e chiarezza dell'analisi.

Consegna: il progetto (codice sorgente) e la relazione finale (.pdf e .tex) devono essere inviati via email all'indirizzo `carolina.crespi@unict.it` entro le ore 14:00 del **17 settembre 2026**. Il candidato è invitato a comunicare la scelta dell'algoritmo da sviluppare entro le ore 16:00 del **5 settembre 2026**.