UNIVERSITÁ DEGLI STUDI DI BRESCIA
Dipartimento di Ingegneria dell'Informazione

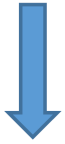
ALNS
Adaptive Large Neighborhood Search

Algoritmi di Ottimizzazione
AA 2016/2017

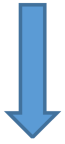
Very Large-Scale Neighborhood Algorithms

- Dimensione del vicinato cresce esponenzialmente al crescere dell'istanza.
- Vicinato troppo grande per essere esplorato in maniera esaustiva.
- Un vicinato più esteso contiene soluzioni migliori.

Metodi tradizionali	Very Large-Scale Neighborhood Search
Applicazione di una mossa piccola (e.g. 2-opt per il TSP)	Applicazione di mosse grandi per vistare intorni più ampi



- Soluzioni molto simili
- Intorni esplorabili rapidamente
- Costo computazionale ridotto

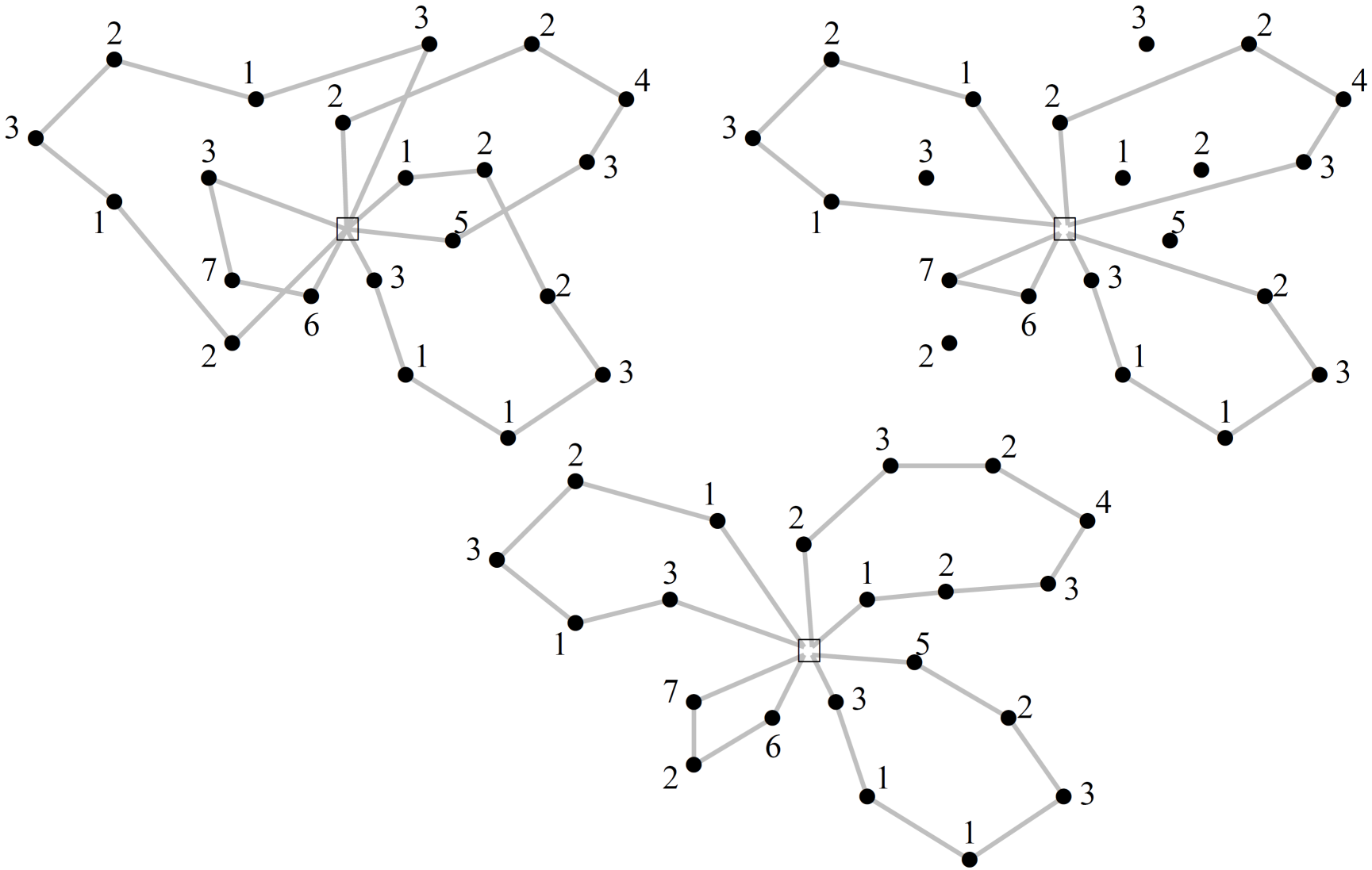


- Diversificazione (30/40% in un singolo step)
- Maggiore qualità
- Costo computazionale elevato

Large Neighborhood Search (LNS)

- Sviluppata da Shaw nel 1997 e applicata inizialmente ai Vehicle Routing Problem.
- Componenti LNS:
  - **Euristica di rimozione (destroy method)**: a partire dalla soluzione corrente e da un numero q, restituisce in output una nuova soluzione dove q componenti della soluzione sono stati rimossi.
  - **Euristica di inserimento (repair method)**: inserisce q elementi nella soluzione prodotta dall'euristica di rimozione, generando una nuova soluzione ammissibile.
- Vicinato definito implicitamente da combinazione di euristica di rimozione e euristica di inserimento.

LNS applicata al VRP



Pseudocodice LNS

Algorithm 1 Large neighborhood search

1: input: a feasible solution x
2: $x^b = x$;
3: **repeat**
4: $x^t = r(d(x))$;
5: **if** accept(x^t, x) **then**
6: $x = x^t$;
7: **end if**
8: **if** $c(x^t) < c(x^b)$ **then**
9: $x^b = x^t$;
10: **end if**
11: **until** stop criterion is met
12: **return** x^b

Il parametro q

- q = dimensione dell'intorno = numero di componenti da rimuovere (grado di distruzione);
- $q \in \{0, \ldots, n\}$, dove:
  - n = numero di componenti nella soluzione;
  - se $q = 0$, nessuna mossa applicata;
  - se $q = n$, risoluzione del problema originario.
- All'aumentare di q, aumenta diversificazione.
  - q basso $\rightarrow$ minimi locali;
  - q elevato $\rightarrow$ riottimizzazione.
- Come gestire q?
  - Incremento graduale;
  - Scelto random in un range predefinito a ogni iterazione.

Ammissibilità delle soluzioni

LNS, in generale, alterna soluzioni infeasible a soluzioni feasible a ogni iterazione.

- Euristica di rimozione genera una soluzione infeasible a partire da una soluzione feasible;
- Euristica di inserimento modifica la soluzione ottenuta per renderla feasible.
- Non è però necessario che la soluzione sia feasible:
  - Si scarta la soluzione e si procede all'iterazione successiva;
  - Penalizzazione dell'infeasibility.

Interpretazione di LNS come *Fix-Optimize*

- **Fix**: un numero di elementi della soluzione è fissato al suo valore corrente (euristica di rimozione);
- **Optimize**: rispettando il vincolo degli elementi fissi, si cerca di ri-ottimizzare la soluzione modificando gli altri parametri (euristica di inserimento).

Da LNS ad ALNS

- LNS usa una sola euristica di rimozione e una di inserimento;
- ALNS invece ha a disposizione un set di euristiche, selezionando una coppia diversa (rimozione, inserimento) a ogni iterazione → Introduzione di un *criterio di selezione* e di un *sistema di valutazione* per le euristiche disponibili.

Pseudocodice ALNS

Algorithm 2 Adaptive large neighborhood search

1: input: a feasible solution x
2: $x^b = x$; $\rho^- = (1,\ldots,1)$; $\rho^+ = (1,\ldots,1)$;
3: **repeat**
4: select destroy and repair methods $d \in \Omega^-$ and $r \in \Omega^+$ using ρ^- and ρ^+;
5: $x^t = r(d(x))$;
6: **if** accept(x^t, x) **then**
7: $x = x^t$;
8: **end if**
9: **if** $c(x^t) < c(x^b)$ **then**
10: $x^b = x^t$;
11: **end if**
12: update ρ^- and ρ^+;
13: **until** stop criterion is met
14: **return** x^b

Euristiche di rimozione

Shaw Removal

Rimuove un elemento in maniera casuale e calcola la *misura di relatività*, i.e. la somiglianza tra l'elemento estratto e gli altri, togliendo poi i $q-1$ più simili.

Esempio (VRP) $\mathcal{R}(i,j) = \frac{1}{c_{ij}+V_{ij}}$

dove c_{ij} = costo per andare da i a j
$V_{ij} = 1$ se i e j sono serviti dallo stesso veicolo.

Worst Removal

Rimuove i q peggiori elementi in termini di costo.

Random Removal

Rimuove q elementi in maniera completamente casuale.

Euristiche di inserimento (I)

Basic Greedy

Inserisce q richieste nelle routes che comportano i costi minori.

Regret

Considera la differenza tra il costo di inserimento di una richiesta nella best route e il costo di inserirla in un'altra.

- Ogni richiesta è valutata considerando k routes e trovando la best route ideale $\rightarrow$ Il valore di *regret* è la somma delle differenze tra il costo minimo nella best route e i costi nelle altre routes.
- Si inserisce per prima la richiesta con il regret più alto e poi si procede in modo iterativo, ricalcolando i valori.
- *Significato del regret*: più è alto, più sarebbe costoso non allocare la richiesta nella sua best route.

Euristiche di inserimento (II)

Hybrid ALNS

- Vengono fissati gli elementi che non sono stati rimossi dall'euristica di rimozione al valore assunto nella soluzione corrente.
- Si utilizza un risolutore MILP per risolvere il sottoproblema così ottenuto.
- Possibilità di aggiungere vincoli alla modellizzazione per:
  - Limitare il numero di cambiamenti rispetto alla soluzione corrente;
  - Garantire che vi sia un numero minimo di cambiamenti rispetto alla soluzione corrente.

Criterio di selezione - *Roulette wheel selection*

- Alternare le euristiche permette di ottenere un metodo più robusto.
- Come scegliere una coppia di euristiche (rimozione, inserimento)? $\rightarrow$ Assegnazione di un peso w_j all'euristica j.
- Date k euristiche di rimozione (o inserimento), con $i \in \{1, \ldots, k\}$, si seleziona l'euristica j con probabilità:

$$\frac{w_j}{\sum_{i=1}^{k} w_i}.$$

- **NB**: l'euristica di rimozione è selezionata *indipendentemente* da quella di inserimento e viceversa.

Aggiornamento dei pesi - *Adaptive Weight Adjustment*

- Come aggiornare i pesi:
  - con una tecnica precisa, sfruttando le statistiche dei risultati ottenuti nelle iterazioni precedenti.
- **IDEA BASE**:
  1. Si suddivide la ricerca e il numero di iterazioni totali in un certo numero di *segmenti* (e.g., un segmento = 100 iterazioni);
  2. All'inizio di ogni segmento, i pesi di tutte le euristiche sono azzerati;
  3. Si assegna un punteggio a ogni euristica ogni volta che è selezionata, in base alla qualità della soluzione calcolata.

Criteri di assegnazione dei punteggi (I)

$$\Psi = \max \begin{cases} \omega_1 & \text{se la nuova soluzione è un nuovo ottimo globale } (x^t > x^b). \\ \omega_2 & \text{se la nuova soluzione è migliore di quella corrente } (x^t > x). \\ \omega_3 & \text{se la nuova soluzione non è migliore di quella corrente ma viene comunque accettata } (x^t \leqslant x \textbf{ and } \text{accept}(x^t, x) = \textbf{true}). \\ \omega_4 & \text{se la nuova soluzione non è migliore di quella corrente e viene rifiutata } (x^t \leqslant x \textbf{ and } \text{accept}(x^t, x) = \textbf{false}). \end{cases}$$

Il peso dell'euristica di rimozione a viene aggiornato secondo la formula

$$\rho_a^- = \lambda \rho_a^- + (1 - \lambda)\Psi$$

dove $\lambda \in [0, 1]$ è detto **parametro di decadimento**

Criteri di assegnazione dei punteggi (II)

Altri aspetti considerabili:

- Pesi normalizzati rispetto all'entità del miglioramento (o peggioramento);
- Pesi normalizzati rispetto al tempo impiegato dall'euristica:
  - Velocità dell'euristica VS qualità della soluzione
- Pesi normalizzati rispetto al numero di iterazioni trascorse dall'ultimo miglioramento di x o x^b
- Favorire euristiche che consentono di diversificare la ricerca

Criterio di accettazione - *Simulated Annealing*

- Data la soluzione corrente x, la soluzione ottenuta x^t e le loro rispettive funzioni obiettivo $c(x)$ e $c(x^t)$, x^t è accettata con probabilità $e^{-\frac{c(x^t)-c(x)}{T}}$;
- T è la temperatura, i.e. una variabile globale tempo-variante, che è impostata alta all'inizio e diminuisce a ogni iterazione:

$$T_{new} = \alpha T_{old}$$

  dove $\alpha \in [0, 1]$ è un parametro che determina la velocità di discesa della temperatura.
  - Possibilità di ridurre T ogni m iterazioni
  - Reset della temperatura se la ricerca è bloccata in minimi locali
- *Simulated Annealing* permette di accettare soluzioni peggiori della soluzione corrente, evitando di bloccarsi in minimi locali.

Criteri di terminazione

- Max numero di iterazioni svolte
- Max iterazioni dall'ultimo miglioramento (di x^b o x)
- Max tempo di computazione

The Rich Pickup and Delivery Problem with Time Windows

- $P = \{1, \ldots, n\}$ nodi di raccolta
- $D = \{n+1, \ldots, 2n\}$ nodi di consegna
- $K = \{1, \ldots, m\}$ veicoli
- P_k, D_k nodi di raccolta e consegna che possono essere serviti dal veicolo k
- Ogni veicolo è associato a un nodo di partenza e un nodo di destinazione (terminale)
- s_i tempo di servizio del nodo i
- $[a_i, b_i]$ finestra temporale in cui va servito il nodo i
- l_i quantità di prodotto che deve essere caricata al nodo i
- C_k capacità del veicolo
- Precedenze tra nodi
- Tutti i clienti (nodo di raccolta + nodo di consegna) devono essere serviti
- Obiettivo: Minimizzazione dei costi di percorrenza

Euristiche di rimozione (I)

Random Removal

Vengono scelte e rimosse q richieste (pickup + delivery) in maniera casuale.

Worst Removal

Vengono rimosse le q richieste con costo maggiore.
Data una soluzione x e una richiesta i, $f'(x, i)$ rappresenta il valore della soluzione se venisse rimossa la richiesta i.
Vengono rimosse iterativamente le q richieste con valore $f(x) - f'(x, i)$ più alto.
Il parametro p controlla la randomizzazione del metodo:

- p basso $\rightarrow$ si rimuovono solo richieste con costo elevato
- p elevato $\rightarrow$ si rimuovono anche richieste meno costose, con probabilità proporzionale a $f(x) - f'(x, i)$ e p.

Euristiche di rimozione (II)

Related Removal

Si seleziona e rimuove una richiesta i in modo casuale. Si calcola la somiglianza r_{ij} con tutte le altre richieste.

$$r_{ij} = \frac{1}{D}(d'(i,j) + d'(i+n,j) + d'(i,j+n) + d'(i+n,j+n))$$

$$d'(u,v) = \begin{cases} d_{uv} & \text{se u o v non sono nodi terminali.} \\ 0 & \text{se u o v sono nodi terminali.} \end{cases}$$

$1 \leqslant D \leqslant 4 =$ numero di termini nell'equazione di r_{ij} diversi da 0 ($D = 1$ nel caso tutti siano pari a 0)

Viene rimossa la richiesta con la somiglianza più alta.
Si seleziona casualmente una delle richieste già rimosse e si rimuove quella più simile.
Si procede fino a che non sono state rimosse q richieste.
Anche in questo caso si introduce una componente random nel metodo.

Euristiche di rimozione (III)

Cluster Removal

Si cercano di rimuovere gruppi di richieste vicine tra di loro.
Si considera una route e, considerando le richieste associate a questa route, si utilizza l'algoritmo di Kruskal (usando r_{ij} come lunghezze degli archi tra richieste). Si termina quando vengono identificate due componenti connesse. Viene scelta casualmente una delle due componenti e vengono rimosse q richieste.
Se meno di q richieste sono state rimosse, si rimuovono altre richieste simili a quelle già rimosse.

Altri esempi di euristiche di rimozione (descritti in [2]) sono:

- Time-oriented removal
- Historical node-pair removal
- Historical request-pair removal

Euristiche di inserimento

Basic Greedy

Si inseriscono le richieste che comportano il minor incremento di costo.
Definito Δf_{ik} l'incremento di costo dovuto all'inserimento della richiesta i nella route k, si selezionano le q richieste con l'incremento di costo minore.

Regret Heuristic

Definito Δf_i^q l'incremento di costo dovuto all'inserimento dell'i-esima richiesta nella q-esima route meno costosa si calcola:

$$\text{(Regret-2)} \qquad i = arg \max_{i \in \{1,\dots,n\}} (\Delta f_i^2 - \Delta f_i^1)$$

e si scelgono le q richieste con valore minore.

$$\text{(Regret-q)} \qquad i = arg \max_{i \in \{1,\dots,n\}} (\sum_{h=2}^{q} \Delta f_i^h - \Delta f_i^1)$$

Applicazioni di ALNS

- *Vehicle Routing Problems* e calcolo di percorsi ottimali (PDPTW, CVRP, VRPTW, SMTWAP, etc.)
- *Arc Routing Problems*
- Problemi di scheduling e pianificazione dei task

Referenze

1. S. Ropke, D. Pisinger (2006), *An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows*, Transportation Science, Vol. 40 (4)
   `http://www.diku.dk/~sropke/Papers/PDPTW_techRep.pdf`.
2. D. Pisinger, S. Ropke (2007), *A general heuristic for vehicle routing problems*, Computers & Operations Research, Vol. 34 (8)
   `http://www.diku.dk/~sropke/Papers/GeneralVRP_TechRep.pdf`.