Ricerca locale

COME OTTIMIZZAZIONE
MA SENZA BOUND

NESSUNA GARANZIA
MATEMATICA

- ▷ La ricerca locale è una tecnica euristica per risolvere problemi di ottimizzazione
- ▷ Non è solitamente garantita terminare in tempo polinomiale
- ▷ Non vi è solitamente alcuna garanzia sulla qualità della soluzione prodotta
- ▷ Solitamente la ricerca locale è molto veloce e produce soluzioni molto buone
- ▷ Nozione di intorno di una soluzione

2-change: intorno di una soluzione

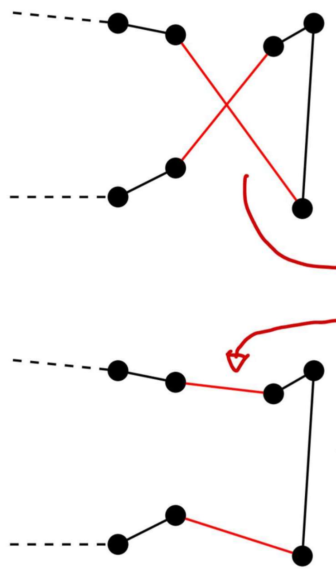
Intorno: 2-change

Sia s una soluzione ammissibile per il problema del commesso viaggiatore. Definiamo l'insieme delle soluzioni nell'intorno di s come

$$I_2(s) = \{t : \ t \text{ differisce da s per 2 archi}\}$$

TSP: 2-Change

PEZZO DI
SOLUZIONE



SOSTITUIBILI CON
ALTRA SOLUZIONE

2 ARCHI TOLTI DAL TOUR
⟹ 1 SOLO MODO
DI RIMPIAZZARLI

2-change

Cardinalità di $I_2(s)$

Se l'istanza del commesso viaggiatore ha n città allora

$$|I_2(s)| = \Theta(n^2)$$

Cardinalità di $S_A(n)$

Sia $S_A(n)$ l'insieme delle soluzioni ammissibili del commesso viaggiatore su n città allora

$$|S_A(n)| = \Theta(n!)$$

$I_2(s)$ vs $S_A(n)$

Se $n = 10$ allora $\frac{|I_2(s)|}{|S_A(n)|} = \frac{n^2}{n!} = 0.000027$

Se $n = 100$ allora $\frac{|I_2(s)|}{|S_A(n)|} = \frac{n^2}{n!} = 1.07 * 10^{-154}$

Se $n = 1000$ allora $\frac{|I_2(s)|}{|S_A(n)|} = \frac{n^2}{n!} = 2.48 * 10^{-2562}$

Ricerca locale

H (n²) VISITABILE

RICERCA-LOCALE(s_{in})

1 $s_{corr} = s_{in}$
2 s_{new} = la migliore soluzione in $I_2(s_{corr})$
3 if $costo(s_{new}) < costo(s_{corr})$
4     $s_{corr} = s_{new}$
5   goto 2
6 else return s_{corr}

PRENDENDO LA MIGLIORE DI TUTTE, ARRIVI PIÚ VELOCEMENTE MA SPENDI DI PIÚ OGNI GIRO

IDEA

- Prendo una soluzione qualunque
- Guardo nell'intorno della soluzione una migliore
- Riapplico fino ad avere la migliore tra gli intorni

OTTIMO LOCALE

Costo computazionale della ricerca locale

Costo computazionale della ricerca locale

Il costo computazionale della ricerca locale dipende da due fattori: dalla dimensione $|I|$ dell'intorno I utilizzato e dal numero m di soluzioni visitate prima di trovare un ottimo locale. Costo $= |I| \cdot m$. Solitamente $|I|$ è facile da calcolare mentre m è quasi sempre non predicibile.

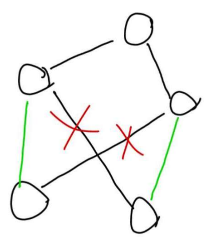
Soluzioni

- ▷ La soluzione trovata dalla ricerca locale dipende dalla soluzione iniziale e dalla sua qualità
- ▷ La soluzione trovata dalla ricerca locale si chiama ottimo locale in quanto è la soluzione migliore nel suo intorno
- ▷ Su architetture parallele può avere senso generare k soluzioni iniziali distinte e applicare in parallelo la ricerca locale partendo dalle k soluzioni iniziali distinte.

Ricerca locale con intorno 2-change

Proprietà geometriche delle soluzioni

Qualunque ottimo locale non presenta incroci !



PESO EUCLIDEO

3-change: intorno di una soluzione

Intorno: 3-change

Sia s una soluzione ammissibile per il problema del commesso viaggiatore. Definiamo l'insieme delle soluzioni nell'intorno di s come

$$I_3(s) = \{t : \; t \text{ differisce da s per} \leq 3 \text{ archi}\}$$

3-change

Cardinalità di $I(s)$

Se l'istanza del commesso viaggiatore ha n città allora

$$|I(s)| = \Theta(n^3)$$

$I_3(s)$ vs $S_A(n)$

Se $n = 10$ allora $\frac{|I_3(s)|}{|S_A(n)|} = \frac{n^3}{n!} = 0.00027$

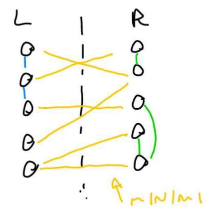
Se $n = 100$ allora $\frac{|I_3(s)|}{|S_A(n)|} = \frac{n^3}{n!} = 1.07 * 10^{-152}$

Se $n = 1000$ allora $\frac{|I_3(s)|}{|S_A(n)|} = \frac{n^3}{n!} = 2.48 * 10^{-2559}$

Partizione bilanciata di grafi

Partizione bilanciata di grafi

Sia $G = (V, E)$ un grafo non orientato e pesato. Trovare una partizione di V in due insiemi L e R tali che la loro cardinalità differisca di al massimo di 1 e la somma dei pesi degli archi che vanno da un nodo di L a un nodo di R sia minima. Il problema così definito è NP-hard.



Metaeuristiche: ricerca locale iterata

```
RICERCA-LOCALE-ITERATA(s_in, n_max)
s_corr = s_in
s_new = RICERCA-LOCALE(s_corr)
n = 0
repeat
    s_jump = PERTURBA(s_corr)
    s_new = RICERCA-LOCALE(s_jump)
    if costo(s_new) < costo(s_corr)
        s_corr = s_new
        n = 0
    else n = n + 1
until n = n_max
return s_corr
```

Swap: intorno di una soluzione

Intorno: Swap

Sia (L, R) una soluzione ammissibile per il problema della partizione bilanciata di grafi. Le soluzioni nel suo intorno le otteniamo scambiando un nodo di L con un nodo di R

Swap

Cardinalità di $I_{swap}(s)$

Se il grafo ha n nodi allora

$$|I_{swap}(s)| = \Theta(n^2)$$

Esempi di Crossover e Mutation per il TSP

Sia $s = \langle c_1, c_2, \ldots, c_n \rangle$ una soluzione per il TSP

PEGGIORE DI RICERCA LOCALE

MUTATION(s)

Scegli due città c_i e c_j in s e scambiale di posizione
return soluzione così ottenuta

Siano s_1 e s_2 due soluzioni per il TSP

CROSSOVER(s_1, s_2)

Scegli un segmento si soluzione in s_1 e completalo con le città mancanti prese da s_2 nell'ordine in cui compaiono in s_2
return soluzione così ottenuta

Algoritmi genetici

BASATI SU EVOLUZIONE DI DARWIN

ALGORITMO-GENETICO($cross, mut$)

$P = \{s_1, s_2, \ldots, s_m\}$
while CONTINUA(P) ES: FINCHÉ MIGLIORA DI ALMENO %
    for $i = 1$ **to** $cross$
        seleziona s_h e s_k
        s_{new} = CROSSOVER(s_h, s_k) CROSSOVER TRA SOLUZIONI
        if TEST(s_{new})
            $P = P \bigcup \{s_{new}\}$ TIENI SE BUONA
    for $i = 1$ **to** mut
        seleziona s_h tra le soluzioni ottenute da crossover
        s_{new} = MUTATION(s_h) MUTA SOLUZIONI
        if TEST(s_{new})
            $P = P \bigcup \{s_{new}\}$ TIENI SE BUONE
return P

NECESSARIO STABILIRE: CROSSOVER, MUTATION, DEF. DI "BUONA"

Esempi di Crossover e Mutation per il TSP

Sia $s = \langle 1, 3, 2, 5, 6, 9, 4, 7, 8, 10 \rangle$ una soluzione per il TSP

MUTATION(s) $= \langle 1, 3, 4, 5, 6, 9, 2, 7, 8, 10 \rangle$ ← CAMBIA 4 ARCHI

Siano $s_1 = \langle 1, 3, 2, 5, 6, 9, 4, 7, 8, 10 \rangle$ e $s_2 = \langle 6, 4, 9, 8, 7, 2, 1, 10, 5, 3 \rangle$ soluzioni per il TSP

CROSSOVER(s_1, s_2) $= \langle 2, 5, 6, 9, 4, 7, 8, 1, 10, 3 \rangle$

MIGLIORAMENTO DOVUTO A TEST NON CROSSOVER