Uso del Machine Learning per Ottimizzare il Processo di Ricerca di un Algoritmo Evolutivo

Simone Basile

Dipartimento di Matematica e Informatica
Università degli Studi di Catania

27 luglio 2023

Sommario

Obiettivo della tesi

Sviluppare un algoritmo evolutivo capace di trovare soluzioni ottimali al problema del Minimum Weighted Feedback Vertex Set

Keywords

- Feedback Vertex Set
- Algoritmi evolutivi
- Machine Learning

Feedback Vertex Set (I)

Introduzione al problema

Il problema del ***Feedback Vertex Set*** (***FVS***) è un noto problema NP-completo nel campo della teoria dei grafi.
In termini semplici, un FVS in un grafo è un insieme di vertici la cui rimozione rende il grafo aciclico. In altre parole, dopo aver rimosso tutti i vertici nel Feedback Vertex Set dal grafo, non dovrebbero rimanere cicli nel sottografo indotto.

Minimum Weighted Feedback Vertex Set

Il ***Minimum Weighted Feedback Vertex Set*** (***MWFVS***) è una variante del problema del Feedback Vertex Set.
Nel problema del MWFVS, ogni vertice nel grafo ha un "peso" associato e l'obiettivo è trovare l'insieme di vertici con il peso complessivo minimo la cui rimozione rende il grafo aciclico.

Feedback Vertex Set (II)

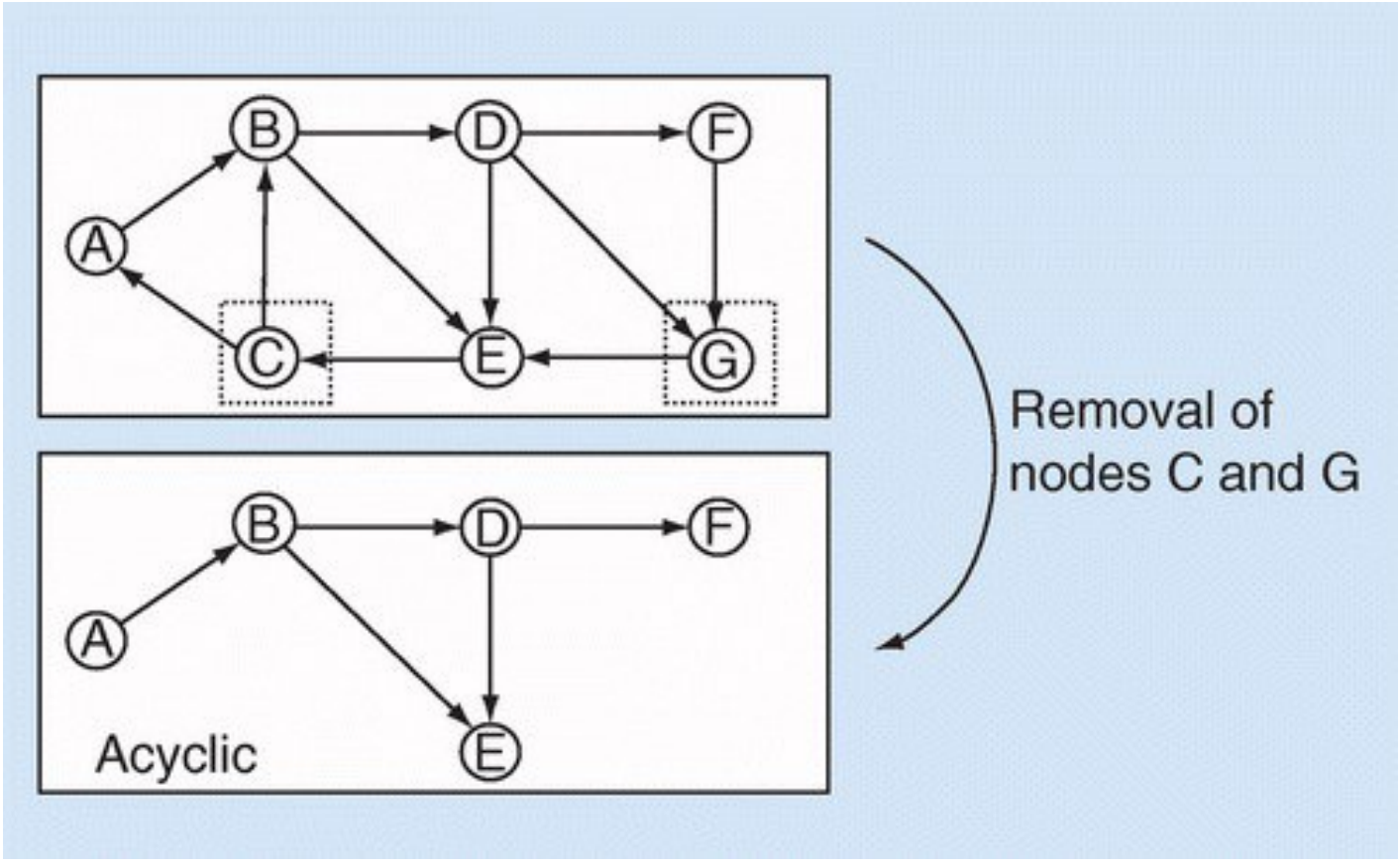


Figure: Esempio funzionamento Feedback Vertex Set

Feedback Vertex Set (III)

Applicazioni pratiche

- **Reti di comunicazione**: dal momento che la presenza di cicli può compromettere la trasmissione dei dati, identificare un FVS ottimale permette di individuare un insieme di nodi critici da rimuovere al fine di garantire una rete priva di cicli
- **Settore dei circuiti integrati**: la presenza di cicli può causare ritardi dei segnali e un consumo energetico eccessivo
- **Sistemi di trasporto**: identificare cicli all'interno di una rete stradale o di trasporto pubblico che possono causare congestione e ritardi
- **Bioinformatica**: identificare un insieme di nodi critici consente di comprendere le dinamiche delle interazioni tra proteine o la regolazione genica

Considerazioni preliminari (I)

Rappresentazione delle soluzioni

Il formato di una generica soluzione all'interno della popolazione è il seguente:

```
0001000000010110100011100000000010
0001010100000011010111000010100 10
0010000001011010001010110001100100 544 1
```

Dove è possibile osservare:

- Una sequenza di n bit in cui il bit posto a 1 indica la rimozione di un nodo dal grafo e l'inserimento di quest'ultimo nel FVS
- Un valore intero rappresentante la fitness della soluzione
- Un bit indicante la validità della soluzione trovata (*feasibility*)

Considerazioni preliminari (II)

Calcolo della fitness e della penalità

La **fitness** rappresenta il "punteggio" assegnato ad una soluzione ed è calcolata come segue:

$$fitness(x) = \sum_{i=0}^{n} x(i) \cdot w(i) \tag{1}$$

Dove $x(i)$ è il bit di presenza dell'i-esimo nodo del grafo nel FVS e $w(i)$ è il peso ad esso associato.
Il valore di **penalità** è stato così definito:

$$\theta = \sum_{i=0}^{n} w(i) \tag{2}$$

Dove $w(i)$ è il peso associato all'i-esimo nodo nel grafo

Considerazioni preliminari (III)

Gestione della penalità

Sulla base delle considerazioni precedenti lo score della soluzione è stato così calcolato:

$$x(n) = \begin{cases} fitness(x) & \text{se il sottografo indotto da } x \text{ è aciclico} \\ \theta + \frac{\theta}{fitness(x)} & \text{altrimenti} \end{cases} \tag{3}$$

Grazie a questo approccio è stato possibile mantenere nella popolazione anche soluzioni unfeasible allo scopo di esplorare al meglio lo spazio delle soluzioni.

Considerazioni preliminari (IV)

Generazione della popolazione iniziale

La **popolazione iniziale** è stata generata dall'applicazione di un approccio stocastico misto ad uno deterministico, in particolare è stata definita una probabilità p con cui un nodo poteva essere inserito nella soluzione:

$$p = \frac{1}{2}\left[(rnd \mod 100) + (\frac{w(i)}{\max w} \cdot 100)\right] \tag{4}$$

Dove rnd indica un valore intero generato casualmente, $w(i)$ rappresenta il peso del nodo i-esimo nel grafo, e $\max w$ si riferisce al peso massimo associato a un nodo all'interno del grafo.

Considerazioni preliminari (V)

Il crossover

Il **crossover** è un'operazione fondamentale negli algoritmi evolutivi in quanto permette di combinare le sequenze di bit di due soluzioni diverse all'interno della popolazione, generando due nuove soluzioni che vengono utilizzate per la generazione successiva.

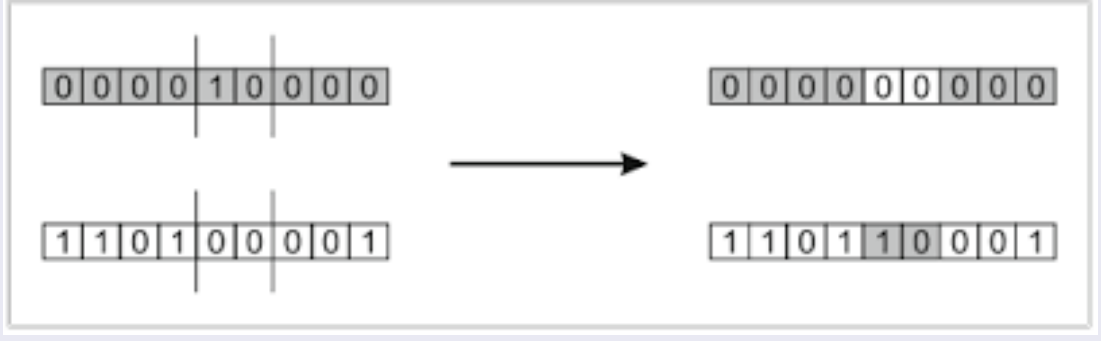


Figure: Esempio funzionamento 2-point crossover

Considerazioni preliminari (VI)

Operatori di crossover in EAML

Inizialmente, sono stati considerati tre tipi di crossover: 3-point, 2-point e 1-point, e ognuno di essi è stato applicato dinamicamente durante le varie iterazioni dell'algoritmo. Tuttavia, dopo analisi e prove sperimentali, è emerso che l'utilizzo esclusivo del crossover di tipo **1-point** ha fornito risultati migliori rispetto agli altri tipi di crossover o alle loro combinazioni.

Ricerca locale

L'algoritmo di ricerca locale

Mira ad alleggerire la soluzione rimuovendo un nodo x avente rapporto peso/grado massimo inserendolo nel sottografo indotto. Da qui seguono due strade:

1. L'inserimento del nodo nel sottografo indotto non ha prodotto cicli
2. L'inserimento del nodo nel sottografo indotto ha prodotto almeno un ciclo

Mentre nel primo caso la ricerca locale conclude con successo, nel secondo caso è necessario ricercare nella componente connessa di x un nodo u di rapporto peso/grado minimo il cui inserimento nel FVS renda il sottografo indotto aciclico. Nel caso in cui non esista un nodo u che soddisfi la condizione precedente, il nodo x viene reinserito in soluzione.

Mutazione (I)

Calcolo della probabilità mutativa

La mutazione rappresenta un operatore che, con una probabilità p (indicata in percentuale), consente la variazione di uno o più bit di una data soluzione. Tale probabilità p è stata calcolata come segue:

$$p(x(i)) = \frac{1}{2}\left[(rnd \mod 100) + (\frac{degree(x(i))}{n} \cdot 100)\right] \quad (5)$$

Dove rnd è un numero generato casualmente e $degree(x(i))$ rappresenta il grado che l'i-esimo nodo assume nel sottografo.

Mutazione (II)

Mutazione: approccio statico

Nella versione iniziale dell'algoritmo (EA), l'approccio alla mutazione era caratterizzato da una natura statica. In particolare, è stata stabilita una soglia mutativa, la quale poteva variare solo all'interno di un insieme predefinito di quattro valori percentuali. Tale variazione avveniva in base alla presenza di stagnazione in un ottimo locale.

La soglia mutativa ($MUTATION\_ODD$) variava come segue: 5% per stagnazione < 30 generazioni, 15% per $30 \leq$ stagnazione < 60 generazioni, 25% per $60 \leq$ stagnazione < 90 generazioni, 50% per stagnazione ≥ 90 generazioni.

Mutazione (III)

Mutazione: approccio Machine Learning (I)

La versione finale dell'algoritmo (EAML) sfrutta un approccio dinamico per definire la soglia mutativa mediante un algoritmo di Machine Learning.

Tale approccio parte da una considerazione geometrica dell'andamento medio della fitness in una finestra di v generazioni. Si consideri la **retta di regressione** della fitness, considerata nella finestra di v generazioni. La funzione che definisce tale retta è:

$$y = \alpha x + \beta \tag{6}$$

Mutazione (IV)

Mutazione: approccio Machine Learning (II)

Assumendo $Y = (Y_1, ..., Y_v))$, il quale rappresenta il vettore contenente la media delle fitness della popolazione in v generazioni, e $X = (X_1, ..., X_v)$, il quale rappresenta il vettore che contiene le v generazioni, e considerando inoltre che σ_X^2 e σ_{XY} sono rispettivamente la varianza di X e la covarianza tra X e Y, i parametri α e β possono essere calcolati come segue:

$$\alpha = \frac{\sigma_{XY}}{\sigma_X^2} = \frac{\sum_{i=1}^{v}(X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^{v}(X_i - \bar{X})^2} \tag{7}$$

$$\beta = \bar{Y} - \alpha\bar{X} \tag{8}$$

dove $\bar{X}$ e $\bar{Y}$ sono rispettivamente le medie di X e Y.

Mutazione (V)

Mutazione: approccio Machine Learning (III)

Considerando che X è un vettore costante $(1, 2, 3, ..., v)$, la sua media e la sua varianza saranno costanti e possono essere definite come:

$$\bar{X} = \frac{1}{v}\sum_{i=1}^{v} i = \frac{1}{v}\frac{v(v+1)}{2} = \frac{1}{2}(v+1) \tag{9}$$

$$\sigma_X^2 = \frac{1}{v}\sum_{i=1}^{v}(i-\bar{X})^2 = \frac{1}{v}\sum_{i=1}^{v} i^2 - \bar{X}^2 \tag{10}$$

Dal momento che $\sum_{i=1}^{v} i^2 = \frac{v^3}{3} + \frac{v^2}{2} + \frac{v}{6}$, possiamo riscrivere l'equazione (10) come:

$$\sigma_X^2 = \frac{v^2}{3} + \frac{v}{2} + \frac{1}{6} - \frac{1}{4}(v+1)^2 = \frac{v^2-1}{12} \tag{11}$$

Mutazione (VI)

Mutazione: approccio Machine Learning (IV)

Sia ϵ un valore soglia molto piccolo ($\epsilon = 0.5$) e sia δ una quantità con cui far crescere/decrescere dinamicamente la soglia mutativa ($\delta = 1$), incontriamo tre casi:

$\alpha > \epsilon$

$$p_m^{(t)} = \left(p_m^{(t-1)} - \delta \right) \Big|_{p_{max}}^{p_{min}} \tag{12}$$

$\alpha < -\epsilon$

$$p_m^{(t)} = \left(p_m^{(t-1)} + \delta \right) \Big|_{p_{max}}^{p_{min}} \tag{13}$$

$|\alpha| < \epsilon$

$$p_m^{(t)} = \left(p_m^{(t-1)} - \frac{\delta}{p_m^f} \right) \Big|_{p_{max}}^{p_{min}} \tag{14}$$

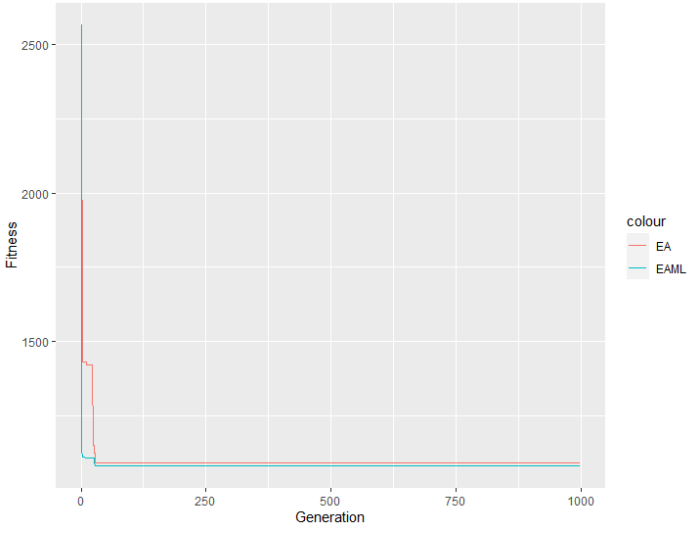
Dove $p_m^{(t)}$ e $p_m^{(t-1)}$ sono rispettivamente la soglia mutativa all'iterazione t e all'iterazione $t-1$, mentre p_m^f ($p_m^f = 0.2$) è un numero intero positivo utilizzato per aumentare la probabilità di una frazione di δ quando la linea di regressione può essere approssimata a una costante ($\alpha \approx 0$). I valori p_{max} e p_{min} sono rispettivamente l'upperbound e il lowerbound della probabilità mutativa.

Comparazione EA vs EAML: Grafi Rand (I)

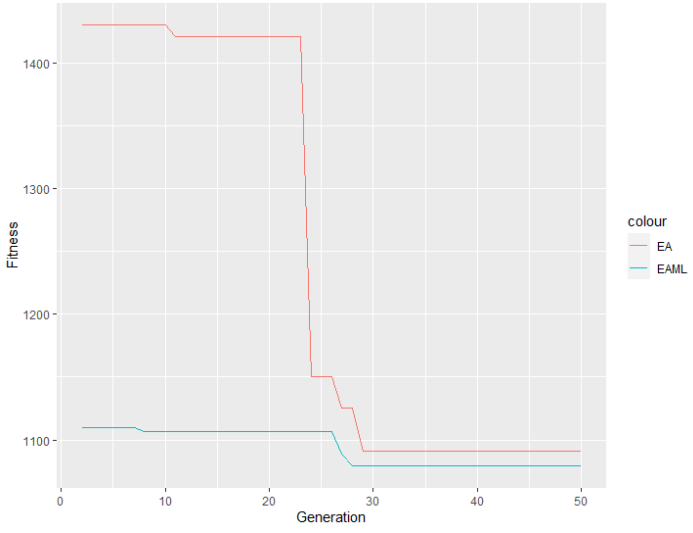
n	e	l	u	EA	EAML	diff
100	247	10	25	507.6	499.4	-8.2
100	247	10	50	851.2	836.4	-14.8
100	247	10	75	1242.6	1213.4	-29.2
100	3069	10	25	1134	1134	0
100	3069	10	50	2179	2179	0
100	3069	10	75	3229.2	3228.6	-0.6
100	841	10	25	831.2	826.8	-4,4
100	841	10	50	1738	1725	-13.0
100	841	10	75	2442	2420.4	-21.6
50	232	10	25	387	386.2	-0.8
50	232	10	50	710.4	709.8	-0.6
50	232	10	75	954.8	951.6	-3.2
50	784	10	25	602.2	602	-0.2
50	784	10	50	1171.8	1172	+0.2
50	784	10	75	1648.8	1648.8	0
50	85	10	25	176	175.4	-0.6
50	85	10	50	283.6	283	-0.6
50	85	10	75	348.2	348	-0.2

Table: Comparazione delle performance degli algoritmi su grafi Rand

Comparazione EA vs EAML: Grafi Rand (II)



(a) Confronto globale tra EA e EAML nel raggiungimento dell'ottimo locale



(b) Ingrandimento andamento algoritmi tra generazione 0 e generazione 50

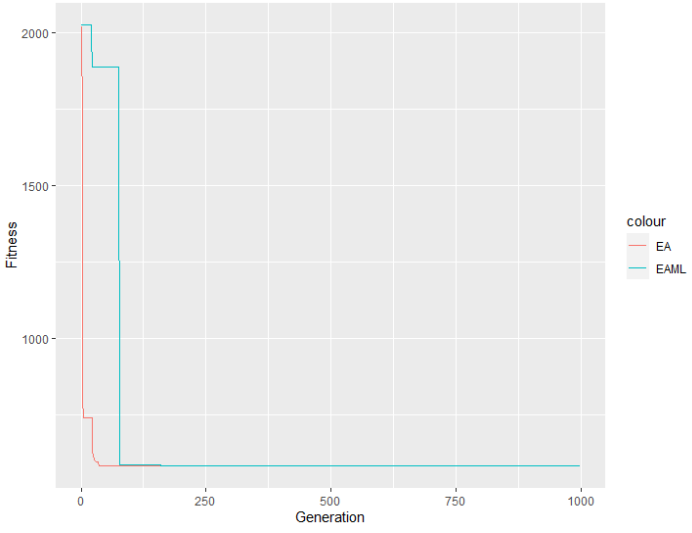
Figure: Grafici di comparazione sull'istanza Rand ***Rand_100_247_7699_10_75*** sulle generazioni

Comparazione EA vs EAML: Grafi Grid (I)

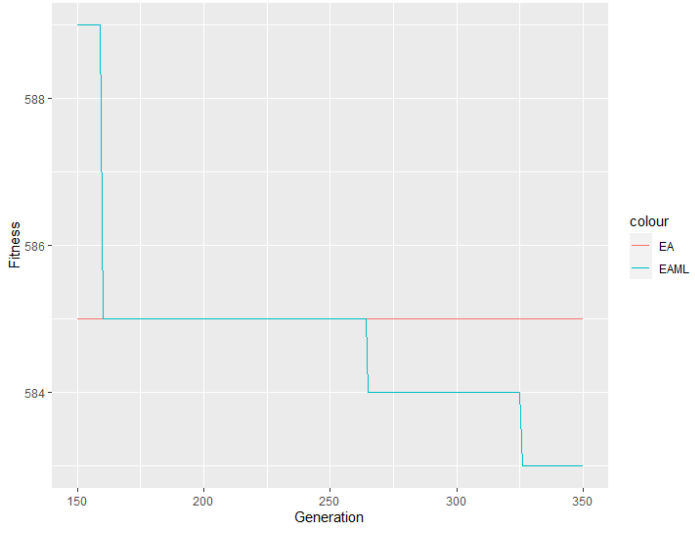
n	e	l	u	EA	EAML	diff
100	180	10	25	575.2	567.6	-7.6
100	180	10	50	967	950.2	-16.8
100	180	10	75	1603	1566.8	-36.2
49	84	10	25	253.8	253.6	-0.2
49	84	10	50	441.4	438.4	-3.0
49	84	10	75	724.4	720.8	-3.6

Table: Comparazione delle performance degli algoritmi su grafi Grid

Comparazione EA vs EAML: Grafi Grid (III)



(a) Confronto globale tra EA e EAML nel raggiungimento dell'ottimo locale



(b) Ingrandimento andamento algoritmi tra generazione 150 e generazione 350

Figure: Grafici di comparazione sull'istanza Grid ***Grid_100_180_1155_10_25*** nel tempo (s)

Conclusioni

Considerazioni finali

Sulla base dei risultati mostrati precedentemente è possbile osservare che:

- L'approccio di Machine Learning ha permesso di migliorare notevolmente la maggior parte dei risultati riportati dalla prima versione dell'algoritmo
- Seppur la convergenza ad una soluzione feasible sia più lenta, la natura dinamica dell'approccio di Machine Learning ha reso possibile più volte l'uscita da stalli di ottimo locale permettendo di trovare soluzioni migliori

Lavori futuri

- Ottimizzazione del metodo di Local Search
- Ricerca di parametri iniziali più efficienti nel tentativo di migliorare le soluzioni ottenute