



UNIVERSITÀ DEGLI STUDI DI CATANIA

DIPARTIMENTO DI MATEMATICA E INFORMATICA

CORSO DI LAUREA MAGISTRALE IN INFORMATICA

Salvatore Alfio Sambataro - Matricola: 1000015834

Applicazione di un algoritmo genetico ad una variante del problema "One Max"

HEURISTICS AND METAHEURISTICS FOR
OPTIMIZATION AND LEARNING
PROGETTO FINALE

Docenti: Prof. Mario Francesco Pavone, Prof. Vincenzo Cutello

Anno Accademico 2023 - 2024

Indice

1	Introduzione	2
1.1	Problema affrontato	2
1.1.1	Formalizzazione del problema	2
1.1.2	Esempio pratico	3
1.1.3	Analisi del problema	3
2	Algoritmo genetico	4
2.1	Algoritmo genetico di base	4
2.2	Miglioramenti: introduzione del machine learning	6
2.2.1	Regressione lineare e aggiornamento dinamico delle probabilità di mutazione e crossover	7
2.2.2	Reinforcement Learning	8
2.2.2.1	Funzionamento del Reinforcement Learning	8
3	Risultati	12
3.1	Qualità delle soluzioni	13
3.1.1	Soluzione migliore, media e deviazione standard di ogni esperimento	13
3.2	Convergenza	21
3.3	Variazione dei parametri dell'algoritmo	27
3.3.1	Variazione dei parametri di crossover e mutazione	27
3.3.2	Variazione delle probabilità di scelta dei metodi di selezione	33
3.3.3	Variazione delle probabilità di scelta dei metodi di selezione (interi esperimenti)	40
3.3.4	Variazione delle probabilità di scelta dei metodi di crossover (RandomGA)	44
3.3.5	Variazione delle probabilità di scelta dei metodi di crossover (interi esperimenti)	50
4	Conclusioni	53
4.1	GA classico	53
4.2	SelectionGA	54
4.3	RandomGA	54
4.4	Considerazioni finali sulla qualità delle soluzioni	54
4.5	Considerazioni finali sui metodi di selezione e crossover	55
Bibliografia		56

Capitolo 1

Introduzione

Il presente documento costituisce la relazione del progetto finale per il corso di "Heuristics and Metaheuristics for optimization and learning".

Il progetto si focalizza sulla risoluzione di un problema di ottimizzazione tramite l'applicazione di un algoritmo genetico. Dopo una breve introduzione al problema specifico scelto, verrà illustrata l'efficacia dell'algoritmo genetico nel trovare soluzioni ottimali o quasi ottimali. L'obiettivo principale è dimostrare come questo approccio metaeuristico, ispirato ai principi dell'evoluzione naturale, possa essere efficacemente utilizzato per affrontare problemi di ottimizzazione, evidenziandone le caratteristiche e i risultati ottenuti.

1.1 Problema affrontato

Il problema affrontato nel presente progetto è una **variante del famoso toy-problem conosciuto come "One-Max"**.

Nella sua versione originale, il problema "One-Max" consiste nel **trovare un vettore binario di lunghezza L che massimizza il numero di bit pari a 1**. In altre parole, l'obiettivo è **trovare una stringa di bit con il maggior numero possibile di 1**.

Nel presente progetto è invece affrontata una variante dell'algoritmo, formulata come segue: dato un vettore $W = (w_1, w_2, \dots, w_L)$ di lunghezza L contenente dei pesi, che possono essere positivi o negativi, l'obiettivo è trovare un vettore di bit X di lunghezza L tale da **massimizzare la somma dei pesi moltiplicati per i bit corrispondenti nelle stesse posizioni**.

1.1.1 Formalizzazione del problema

Il problema, che nel seguito sarà indicato con "**Weighted One-Max**", può essere formulato dal punto di vista "algoritmico" come segue:

- **Input:**
 - lunghezza L del vettore
 - vettore dei pesi $W = (w_1, w_2, \dots, w_L), w_i \in \mathbb{R}$

- **Output:**

- vettore di bit $X = (x_1, x_2, \dots, x_L)$ di lunghezza L che massimizza la seguente quantità:

$$\sum_{i=1}^L w_i \cdot x_i$$

1.1.2 Esempio pratico

Supponiamo che $L = 5$ e $W = (4, -2, 7, -6, 3)$. Alcune possibili soluzioni al problema potrebbero essere le seguenti:

- $X_1 = (1, 1, 1, 0, 0) \rightarrow \sum_{i=1}^L w_i \cdot x_i = 9$
- $X_2 = (0, 1, 1, 1, 0) \rightarrow \sum_{i=1}^L w_i \cdot x_i = -1$
- $X_3 = (1, 0, 1, 0, 0) \rightarrow \sum_{i=1}^L w_i \cdot x_i = 11$

In questo caso, la soluzione migliore è chiaramente X_3 .

1.1.3 Analisi del problema

Più in generale, è possibile intuire che **le soluzioni ottimali sono quelle che presentano allo stesso tempo:**

- il maggior numero possibile di bit a 1 in corrispondenza di pesi positivi
- il maggior numero possibile di bit a 0 in corrispondenza di pesi negativi

La **soluzione ottima globalmente** è quindi quella che presenta:

- 1 in corrispondenza di ognuno dei pesi positivi
- 0 in corrispondenza di ognuno dei pesi negativi

Nell'esempio precedente la **soluzione ottima** sarebbe quindi $X^* = (1, 0, 1, 0, 1) = 14$

Capitolo 2

Algoritmo genetico

La tipologia di algoritmi scelta per la risoluzione del problema proposto è la classe degli **algoritmi genetici**.

La scelta di utilizzare un algoritmo genetico ("GA") per risolvere il problema è motivata dalla sua capacità di esplorare efficacemente spazi di soluzione complessi in tempi di esecuzione ragionevoli. Gli algoritmi genetici, essendo degli algoritmi basati su metaeuristiche, sono particolarmente adatti per problemi in cui la funzione obiettivo presenta molteplici picchi e avvallamenti, caratteristiche che rendono difficolta l'applicazione di metodi di ottimizzazione tradizionali. Inoltre, la natura stocastica degli algoritmi genetici consente di evitare la convergenza prematura verso ottimi locali, migliorando le possibilità di individuare la soluzione ottima globalmente.

2.1 Algoritmo genetico di base

Un algoritmo genetico è un metodo di ricerca euristica ispirato ai processi di evoluzione naturale. L'algoritmo genetico di base può essere descritto attraverso le seguenti fasi:

- **Inizializzazione:** Una popolazione iniziale di soluzioni candidate viene generata casualmente. La rappresentazione degli individui dipende dal problema affrontato: nel nostro caso, gli individui saranno rappresentati da un vettore binario di lunghezza L .
- **Valutazione:** Ogni soluzione nella popolazione viene valutata utilizzando una "funzione di fitness", che nel nostro caso è data da:

$$f(X, W) = \sum_{i=1}^L w_i \cdot x_i$$

- **Selezione:** Gli individui vengono selezionati per la riproduzione, solitamente in base alla loro fitness. Tecniche comuni includono la selezione proporzionale alla fitness e il torneo.
- **Crossover:** Coppie di individui della popolazione selezionati combinano i loro "geni" per creare nuovi figli. Questo è un processo, insieme alla mutazione, che serve ad introdurre variazione nella popolazione, al fine di esplorare efficacemente lo spazio delle soluzioni. Alcuni esempi sono Single Point Crossover e K-Point Crossover.
Il crossover tra due individui avviene con una certa probabilità, che prende il nome di **Crossover Rate**, il cui valore è compreso tra 0 e 1.

- **Mutazione:** Alcuni individui della popolazione subiscono mutazioni casuali, che alterano uno o più dei loro geni. La mutazione aiuta a mantenere la diversità genetica nella popolazione.
La mutazione di un individuo avviene con una certa probabilità, che prende il nome di **Mutation Rate**, il cui valore è compreso tra 0 e 1.
- **Sostituzione:** La nuova generazione di individui sostituisce la vecchia popolazione.
- Si ripete il processo evolutivo in maniera iterativa, fino al raggiungimento di un criterio di terminazione, come un numero massimo di generazioni, una soglia di fitness desiderata, un numero di calcoli massimo della funzione di fitness, e così via.

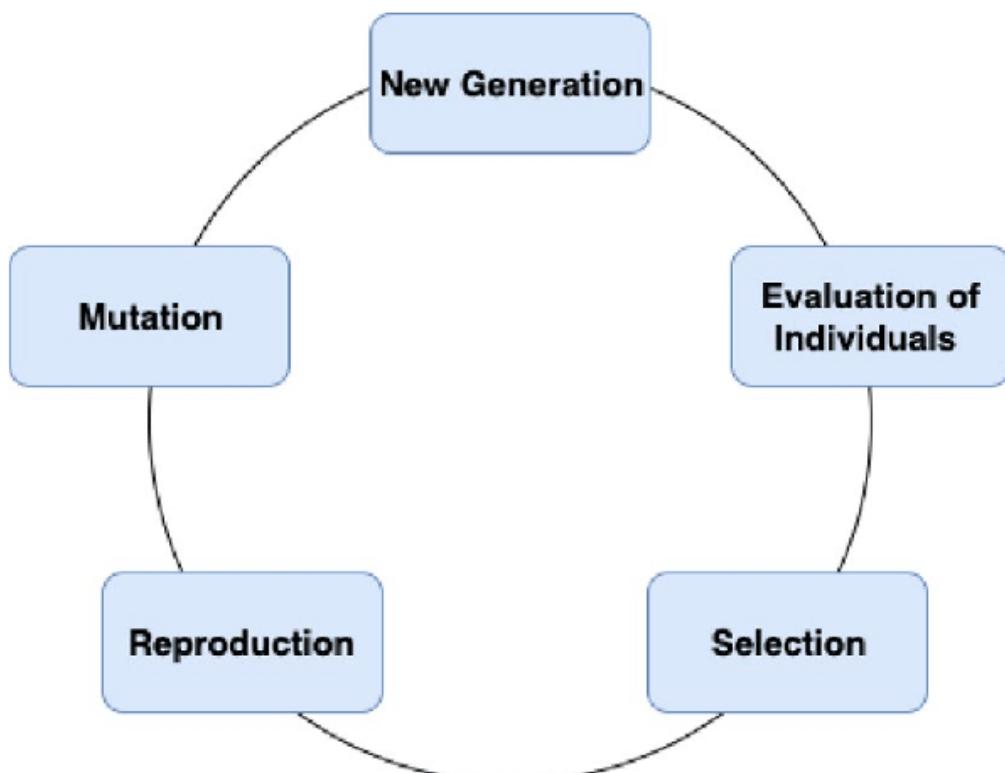


Figura 2.1: Ciclo di un algoritmo genetico

2.2 Miglioramenti: introduzione del machine learning

Per migliorare l'efficacia dell'algoritmo genetico, è stato scelto di integrare una componente di **machine learning** al suo interno. In particolare, l'idea è quella di utilizzare l'algoritmo genetico per la ricerca delle soluzioni, ed usare il machine learning per **ottimizzare la scelta dei parametri e dei metodi di selezione e crossover usati dall'algoritmo**.

La metodologia usata unisce due tecniche proprie del Machine Learning: **regressione lineare** e un qualcosa di simile al concetto di **Reinforcement Learning**.

I due approcci sono implementati come segue:

- **Regressione lineare**: usata per **predire l'andamento della qualità delle soluzioni trovate dall'algoritmo e modificare dinamicamente i parametri relativi alle probabilità di crossover e di mutazione.** [1]
- **Reinforcement Learning**: viene usato per **modificare dinamicamente il metodo di selezione e di crossover usato ad ogni iterazione dell'algoritmo genetico**. In particolare, sulla base delle predizioni della goodness della popolazione, effettuate tramite regressione lineare, viene assegnato un "premio" oppure una "penalità" a ciascuno dei possibili metodi, sulla base del fatto che il metodo abbia migliorato o meno l'andamento previsto della popolazione precedente.

Saranno presentate in particolare due varianti dell'algoritmo genetico:

- **"SelectionGA"**: sulla base della regressione lineare e del Reinforcement learning, l'algoritmo cambia dinamicamente **crossover rate, mutation rate** e il **metodo di selezione degli individui**. La scelta del metodo di selezione può avvenire tra **quattro metodi**:
 - **Tournament Selection**
 - **Roulette Selection**
 - **Elitism Selection**
 - **Rank Selection**
- **"RandomGA"**: oltre a cambiare crossover rate, mutation rate e metodo di selezione degli individui tra cui effettuare crossover, in questa versione viene modificato dinamicamente anche il **metodo di cross-over usato per generare nuovi individui**. I metodi di cross-over considerati sono:
 - **Single-point crossover**
 - **K-point crossover**
 - **Uniform crossover**

2.2.1 Regressione lineare e aggiornamento dinamico delle probabilità di mutazione e crossover

La regressione lineare è un modello predittivo, propria del machine learning, che permette di **prevedere valori che si osserveranno in futuro, sulla base di valori "passati"**.

In questo caso, la regressione è usata per predire la goodness della popolazione, sulla base della fitness media della popolazione, in un certo numero v di iterazioni "precedenti".

Dal punto di vista matematico, la regressione lineare può essere formalizzata come:

$$y = \alpha x + \beta$$

La precedente equazione rappresenta l'equazione di una retta nel piano, e in particolare **il parametro α permette di studiare l'inclinazione della retta**, ovvero nel nostro caso *l'andamento della goodness della popolazione, sulla base dei valori di fitness degli individui presenti in essa*.

In particolare:

- $\alpha > 0$: la goodness della popolazione cresce
- $\alpha < 0$: la goodness della popolazione diminuisce
- $\alpha = 0$: la goodness della popolazione rimane pressochè la stessa

Per il presente progetto, la regressione viene calcolata con il metodo **OLS (Ordinary Least Squares)**.

Nello specifico, si ha che:

- $\alpha = \frac{\sigma_{XY}}{\sigma_X^2} = \frac{\sum_{i=1}^v (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^v (X_i - \bar{X})^2}$, con $X = (1, \dots, v)$ e $Y = (Y_1, Y_2, \dots, Y_v)$ vettore contenente la fitness media della popolazione in ognuna delle v generazioni
- $\beta = \bar{Y} - \alpha \bar{X}$

L'aggiornamento dei parametri di crossover e mutazione è fatta come proposto in Cavallaro et al. "GAML" [1].

2.2.2 Reinforcement Learning

Il Reinforcement Learning è una tecnica di Machine Learning in cui un computer (“**agente**”) impara a svolgere un’attività tramite ripetute interazioni di tipo “trial-and-error” con un ambiente dinamico. Questo approccio all’apprendimento consente all’agente di adottare una serie di decisioni in grado di massimizzare una metrica di ”**ricompensa**” per l’attività, senza essere esplicitamente programmato per tale operazione e senza l’intervento del programmatore.

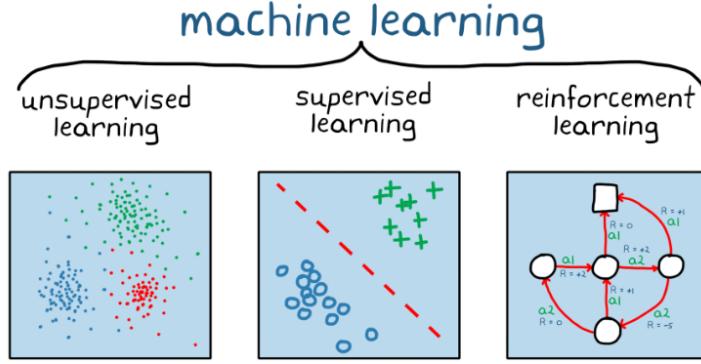


Figura 2.2: Confronto tra Machine Learning classico e Reinforcement Learning

2.2.2.1 Funzionamento del Reinforcement Learning

I concetti principali su cui si basa un algoritmo di Reinforcement Learning sono i seguenti:

- **Agente:** L’agente è l’entità che interagisce con l’ambiente, prende decisioni basate sugli stati osservati e apprende dalle ricompense ricevute. L’agente utilizza una policy per determinare quali azioni compiere in base agli stati attuali.
- **Ambiente:** l’ambiente è tutto ciò che interagisce con l’agente e che viene influenzato dalle azioni di quest’ultimo. Nel nostro caso, l’ambiente è rappresentato dall’insieme dei dati coinvolti nell’algoritmo genetico, ad esempio la popolazione, le probabilità di crossover e mutazione, i metodi di selezione/crossover, la dimensione della popolazione, e così via...
- **Stato S_t :** lo stato rappresenta una configurazione specifica del sistema o dell’ambiente in un dato momento. Esso contiene tutte le informazioni necessarie per descrivere completamente la situazione corrente dell’agente nell’ambiente.
In questo caso, lo stato comprende informazioni quali le probabilità associate ai vari metodi di selezione e crossover e le probabilità di mutazione e crossover, popolazione attuale, e così via...
- **Azione A_t :** L’azione è una decisione che l’agente può compiere in uno stato specifico. L’insieme di tutte le azioni possibili in uno stato definisce lo ”spazio delle azioni”. Nel nostro caso, lo spazio delle azioni consiste, ad esempio, nell’uso di un metodo di selezione/crossover piuttosto che un’altro.

- **Ricompensa** R_t : La ricompensa è un feedback ricevuto dall'agente per aver scelto un determinato metodo di selezione/crossover.

In questo caso, la ricompensa consiste in un **incremento della probabilità di selezionare un metodo di selezione/crossover specifico** se la goodness della popolazione migliora usando quello specifico metodo. In questo modo, se la goodness migliora dopo l'applicazione di un certo metodo di selezione/crossover, l'agente riceve una ricompensa positiva, che aumenta la probabilità di riutilizzare quel metodo in futuro.

- **Policy** π : la policy rappresenta la strategia o l'insieme di regole che l'algoritmo segue per prendere decisioni. E' solitamente basata sulle reward accumulate dall'agente. In questo caso, la policy è basata sull'andamento della goodness della popolazione e su un parametro Δ , e l'incremento assegnato come Reward ad un metodo che ha portato ad un miglioramento della goodness consiste semplicemente nell'**incrementare di un fattore** Δ la probabilità attuale del metodo scelto, per poi **normalizzare tra 0 e 1 le nuove probabilità ottenute**.

Il funzionamento dell'algoritmo di Reinforcement Learning applicato all'ottimizzazione dei parametri di un algoritmo genetico è il seguente:

1. **Inizializzazione dell'ambiente**: l'agente parte con una probabilità iniziale uguale per ogni metodo di selezione/crossover.
2. **Osservazione dello Stato Corrente**: L'agente osserva lo stato corrente, che include le probabilità correnti per ogni metodo di selezione/crossover e la fitness media della popolazione.
3. **Selezione dell'Azione (Metodo di Selezione/Crossover)**: basandosi sulle probabilità attuali, l'agente seleziona un metodo di selezione/crossover da applicare.
4. **Esecuzione dell'Azione**: viene applicato il metodo di selezione/crossover scelto per generare una nuova popolazione
5. **Aggiornamento della Popolazione e Valutazione della Fitness**: La nuova popolazione di figli sostituisce la popolazione corrente e viene valutata la goodness della nuova popolazione.
6. **Reward e aggiornamento dello stato**: Se la goodness della popolazione è migliorata, l'agente riceve una ricompensa positiva, che aumenta la probabilità di scelta del metodo di selezione/crossover che è stato usato. In caso contrario, la probabilità può essere ridotta. Lo stato viene aggiornato con le nuove probabilità di selezione per i metodi di selezione/crossover e la nuova goodness della popolazione.
7. **Ripetizione**: Il processo continua iterativamente fino al raggiungimento di un criterio di arresto (in questo caso, un **numero massimo di generazioni** oppure un **numero di calcoli della fitness al più pari a 10^6**).

Di seguito sono riportati gli pseudocodici degli algoritmi SelectionGA e RandomGA:

Algorithm 1: SelectionGA

Data:

k : generations
 n : population size
 v : regression buffer size
 ϵ : regression goodness threshold
 Δ : reward
 p_c^f : crossover probability factor
 p_c^l : crossover probability level
 p_m^f : mutation probability factor
 p_m^l : mutation probability level
 $weights$: vector of weights $W = (w_1, \dots, w_L)$, $w_i \in \mathbb{R}$
 $T_{max} = 10^6$: # max. fitness evaluations
 p_{max}^c : maximum crossover probability
 p_{min}^c : minimum crossover probability
 p_{max}^m : maximum mutation probability
 p_{min}^m : minimum mutation probability

Result:

best solution
crossover rates
mutation rates
selection methods rates

```
 $t \leftarrow 0;$ 
Initialize an empty circular buffer  $H$  with a capacity of  $v$ ;
 $P^{(t)} \leftarrow initialize\_population(n, len(weights));$ 
 $p_c \leftarrow 0.5;$ 
 $p_m \leftarrow 0.01;$ 
 $prob\_selection \leftarrow [0.25, 0.25, 0.25, 0.25];$ 
 $method\_selection \leftarrow "tournament";$ 
while  $t \leq k$  AND  $t_{fit} \leq T_{max}$  do
     $probs\_selection, method\_selection, P^* \leftarrow$ 
         $optimize\_selection(probs\_selection, weights, H, \epsilon, P, method\_selection, \Delta);$ 
     $p_c, p_m \leftarrow optimize\_parameters(H, \epsilon, p_c^f, p_c^l, p_m^f, p_m^l);$ 
     $P^* \leftarrow crossover\_operator(P^*, p_c);$ 
     $P^* \leftarrow mutation\_operator(P^*, p_m);$ 
     $avg\_fit \leftarrow average\_fitness(P^*, weights);$ 
     $append(H, average\_fitness);$ 
     $P \leftarrow P^*;$ 
     $t \leftarrow t + 1;$ 
end
return best individual in  $P_t$ , crossover_rates ,mutation_rates, selection_method_rates
```

Algorithm 2: RandomGA

Data:

k : generations
 n : population size
 v : regression buffer size
 ϵ : regression goodness threshold
 Δ : reward
 p_c^f : crossover probability factor
 p_c^l : crossover probability level
 p_m^f : mutation probability factor
 p_m^l : mutation probability level
 $weights$: vector of weights $W = (w_1, \dots, w_L)$, $w_i \in \mathbb{R}$
 $T_{max} = 10^6$: # max. fitness evaluations
 p_{max}^c : maximum crossover probability
 p_{min}^c : minimum crossover probability
 p_{max}^m : maximum mutation probability
 p_{min}^m : minimum mutation probability

Result:

best solution
crossover rates
mutation rates
selection methods rates
crossover methods rates

$t \leftarrow 0$;
Initialize an empty circular buffer H with a capacity of v ;
 $P^{(t)} \leftarrow initialize_population(n, len(weights))$;
 $p_c \leftarrow 0.5$;
 $p_m \leftarrow 0.01$;
 $prob_selection \leftarrow [0.25, 0.25, 0.25, 0.25]$;
 $prob_crossover \leftarrow [0.33, 0.33, 0.34]$;
 $method_selection \leftarrow "tournament"$;
 $method_crossover \leftarrow "single_point"$;
while $t \leq k$ AND $t_{fit} \leq T_{max}$ **do**
 $probs_selection, method_selection, P^* \leftarrow$
 $optimize_selection(probs_selection, weights, H, \epsilon, P, method_selection, \Delta)$;
 $p_c, p_m \leftarrow optimize_parameters(H, \epsilon, p_c^f, p_c^l, p_m^f, p_m^l)$;
 $probs_crossover, method_crossover, P^* \leftarrow$
 $optimize_crossover(probs_crossover, weights, H, \epsilon, P^*, method_crossover, \Delta)$;
 $P^* \leftarrow mutation_operator(P^*, p_m)$;
 $avg_fit \leftarrow average_fitness(P^*, weights)$;
 $append(H, average_fitness)$ $P \leftarrow P^*$ $t \leftarrow t + 1$
end
return best individual in P_t , crossover_rates, mutation_rates, selection_method_rates, crossover_method_rates

Capitolo 3

Risultati

In questo capitolo sono riportati i risultati dell'esecuzione dell'algoritmo su diverse istanze, differenti tra loro per la dimensione del vettore dei pesi e i possibili valori assunti da questi ultimi. Le implementazioni dei 3 algoritmi confrontati, ovvero GA classico, SelectionGA e RandomGA, sono state scritte in **linguaggio Python**, e sono allegate alla presente relazione. Sono in particolare riportate analisi che riguardano diversi aspetti dell'algoritmo, tra cui:

- **Qualità della soluzione** (media e deviazione standard delle soluzioni trovate, confronto tra soluzione ottima e soluzione migliore trovata)
- **Convergenza dell'algoritmo**
- **Andamento delle variazioni dei parametri significativi dell'algoritmo**

Per quanto riguarda le istanze su cui è stato valutato l'algoritmo, sono proposte le seguenti **tre tipologie di istanze**:

- Vettore dei pesi di dimensione 500
- Vettore dei pesi di dimensione 1000
- Vettore dei pesi di dimensione 1500

A sua volta, ogni istanza è suddivisa in tre categorie, sulla base dei possibili valori dei pesi:

- Valori dei pesi compresi tra -5 e 5
- Valori dei pesi compresi tra -10 e 10
- Valori dei pesi compresi tra -25 e 25

Si è quindi testato l'algoritmo su un totale di **9 tipologie di "esperimenti"**. Ogni esperimento è stato eseguito 50 volte in maniera **indipendente**, ovvero ogni volta è stato cambiato il seme del generatore dei numeri casuali dell'algoritmo. Si è inoltre posto un **limite al numero di calcoli massimi per la funzione di fitness**, pari a 10^6 .

Per ognuna delle diverse analisi proposte saranno confrontati tre algoritmi: GA classico, SelectionGA e RandomGA

3.1 Qualità delle soluzioni

Sono di seguito riportate delle analisi riguardanti la qualità delle soluzioni trovate nei diversi esperimenti.

3.1.1 Soluzione migliore, media e deviazione standard di ogni esperimento

	GA	SelectionGA	RandomGA
Optimal		672.8000000000001	
Best	671.22	672.8000000000001	672.8000000000001
Mean	665.5148000000002	672.8000000000001	672.8000000000001
Std.Dev.	3.7925338442787835	1.13e-13	1.12e-13
Time (s)	344	478	965

Tabella 3.1: Valori relativi alla qualità della soluzione per l'istanza: 500_-5_to_+5

	GA	SelectionGA	RandomGA
Optimal		1345.44	
Best	1345.44	1345.44	1345.44
Mean	1333.030399999999	1345.44	1345.44
Std.Dev.	8.130913100015281	2.27e-13	2.25e-13
Time (s)	366	485	1103

Tabella 3.2: Valori relativi alla qualità della soluzione per l'istanza: 500_-10_to_+10

	GA	SelectionGA	RandomGA
Optimal		3363.450000000003	
Best	3360.720000000003	3363.45	3363.45
Mean	3331.7594	3363.440000000007	3363.45
Std.Dev.	15.483944253322523	9.09e-13	9.07e-13
Time (s)	344	490	1074

Tabella 3.3: Valori relativi alla qualità della soluzione per l'istanza: 500_-25_to_+25

	GA	SelectionGA	RandomGA
Optimal		1248.42	
Best	1150.889999999999	1247.03	1248.42
Mean	1088.3142	1243.052199999999	1248.4156
Std.Dev.	17.02185302368692	2.2285154610188043	0.020214846029570858
Time (s)	389	531	1232

Tabella 3.4: Valori relativi alla qualità della soluzione per l'istanza: 1000_-5_to_+5

	GA	SelectionGA	RandomGA
Optimal		2496.74	
Best	2255.98	2492.7	2496.74
Mean	2166.6028	2486.516	2496.7302
Std.Dev.	38.424575731685024	4.866582373699227	0.04818672016226943
Time (s)	379	512	1260

Tabella 3.5: Valori relativi alla qualità della soluzione per l'istanza: 1000_-10_to_+10

	GA	SelectionGA	RandomGA
Optimal		6241.610000000001	
Best	5574.030000000001	6233.51	6241.61
Mean	5433.2212	6215.052200000001	6241.6096
Std.Dev.	71.9329794639426	10.005837554148107	0.002799999999933789
Time (s)	383	510	1212

Tabella 3.6: Valori relativi alla qualità della soluzione per l'istanza: 1000_-25_to_+25

	GA	SelectionGA	RandomGA
Optimal		1851.37	
Best	1401.179999999998	1771.259999999998	1851.179999999998
Mean	1339.4194	1733.0472	1850.08
Std.Dev.	25.31113600058281	21.95918669167871	0.8750977088302715
Time (s)	473	555	1785

Tabella 3.7: Valori relativi alla qualità della soluzione per l'istanza: 1500_-5_to_+5

	GA	SelectionGA	RandomGA
Optimal		3786.13	
Best	2904.029999999997	3617.34	3785.9199999999996
Mean	2750.914199999997	3552.748799999994	3783.149799999997
Std.Dev.	60.40784469222519	31.523136750012682	2.0876517813083066
Time (s)	479	568	2389

Tabella 3.8: Valori relativi alla qualità della soluzione per l'istanza: 1500_-10_to_+10

	GA	SelectionGA	RandomGA
Optimal		9465.59	
Best	7176.08	9081.86	9465.1
Mean	6937.2036	8903.943200000002	9459.253600000002
Std.Dev.	121.84414047068495	76.24981241786766	6.011769376814245
Time (s)	447	575	2408

Tabella 3.9: Valori relativi alla qualità della soluzione per l'istanza: 1500_-25_to_+25

Dalle tabelle riportate in precedenza, si evince che, per il problema preso in esame:

- l'algoritmo GA classico funziona abbastanza bene quando la dimensione del vettore dei pesi e la variabilità di questi ultimi non sono molto elevate (ad esempio, nell'istanza *500_-5_to_+5*). Il vantaggio di questa versione dell'algoritmo GA è inoltre quella di avere tempi di esecuzione significativamente inferiori rispetto alle altre versioni, con un risparmio pari a circa il 20% rispetto a SelectionGA e del 60-65% rispetto a RandomGA. Tuttavia, negli esperimenti con un vettore dei pesi più lungo e con un range di pesi più ampio, l'algoritmo GA classico non restituisce buoni risultati in termini di qualità della soluzione, arrivando ad avere nell'esperimento *1500_-25_to_+25* un errore pari a circa il 24% sul valore della soluzione migliore trovata rispetto a quella ottimale, e un errore in media pari a circa 27%.
- l'algoritmo SelectionGA, con un vettore dei pesi con dimensioni non troppo elevate (500 e 1000) riesce a trovare la soluzione ottima nella maggior parte dei casi, mentre quando il vettore dei pesi è più esteso, SelectionGA non sempre riesce a trovare la soluzione ottima, ma riesce comunque a trovare una soluzione che vi si avvicina molto, con un errore della migliore soluzione rispetto a quella ottimale pari a circa il 4%.
Dal punto di vista dell'efficienza, i tempi di esecuzione sono di poco superiori rispetto a GA classico, ma sono allo stesso tempo molto più bassi rispetto a RandomGA.
- l'algoritmo RandomGA ottiene risultati pari all'ottimo per le istanze di dimensione 500 e 1000, mentre ottiene risultati subottimali nelle istanze di lunghezza 1500, le quali sono comunque estremamente vicine alla soluzione ottima, con un errore in questo ultimo caso pari a circa il 0.1% tra soluzione migliore trovata e soluzione ottima.
Il problema principale di questo algoritmo sono i tempi di esecuzione: come si può notare infatti, a parità di istanza RandomGA impiega in media circa 4 volte il tempo impiegato da SelectionGA e circa 5 volte il tempo impiegato da GA classico.

Fatte le considerazioni precedenti, possiamo concludere che:

- GA classico garantisce maggiore efficienza computazionale, e funziona relativamente bene con istanze del problema non troppo complesse. Non riesce invece a raggiungere buoni risultati riguardo le istanze più complesse del problema.
Può quindi essere usato quando non sono richieste soluzioni estremamente vicine all'ottimo, e allo stesso tempo è richiesta un'elevata efficienza computazionale
- SelectionGA è una via di mezzo tra GA classico e RandomGA, in quanto permette di trovare soluzioni ottime nelle istanze meno complesse, e raggiunge comunque buoni risultati anche per le istanze più complesse, il tutto con un tempo di esecuzione di poco superiore rispetto a GA classico ma molto inferiore rispetto a RandomGA
- RandomGA è, dal punto di vista della qualità delle soluzioni, il migliore dei 3 algoritmi proposti, in quanto riesce a raggiungere nella totalità dei casi una soluzione pari o comunque estremamente vicina all'ottimo. Lo svantaggio principale sono però i lunghi tempi di esecuzione.
Questo algoritmo può quindi essere usato quando è necessario trovare soluzioni quanto più vicine possibile all'ottimo, e allo stesso tempo l'efficienza computazionale non è un requisito vincolante.

Per maggiore chiarezza sulle soluzioni trovate nei diversi esperimenti, sono di seguito riportati degli scatter-plot relativi alle migliori soluzioni trovate dall'algoritmo in ognuna delle 50 run, in **alcune** delle istanze proposte.

Per ognuna delle istanze sono riportate all'interno del grafico:

- **Variazione della fitness (asse y) della soluzione migliore trovata nelle diverse run (asse x)**
- **Media delle migliori soluzioni trovate (linea rossa)**
- **Deviazione standard delle migliori soluzioni trovate (linee verdi)**

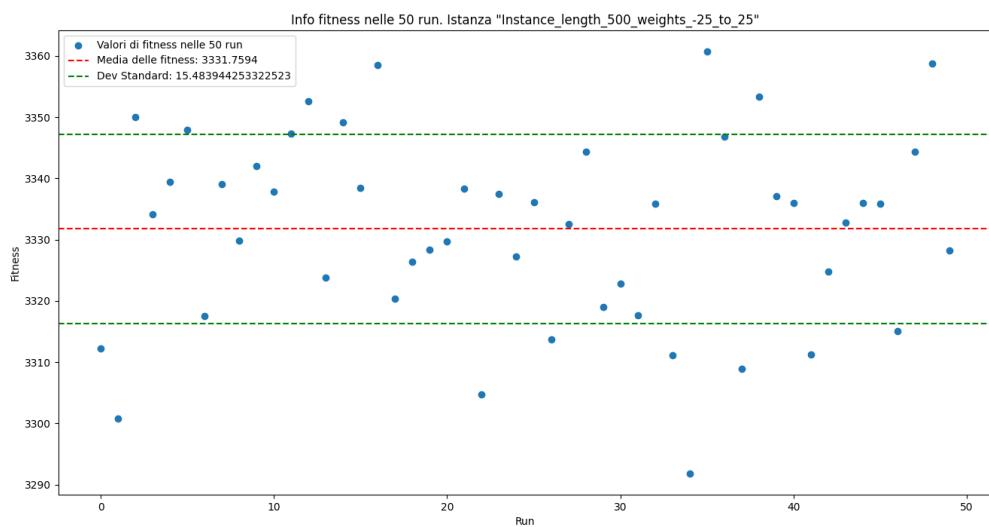


Figura 3.1: Scatterplot delle migliori soluzioni delle 50 run:

Istanza: 500_-25_to_+25

Algoritmo: GA classico

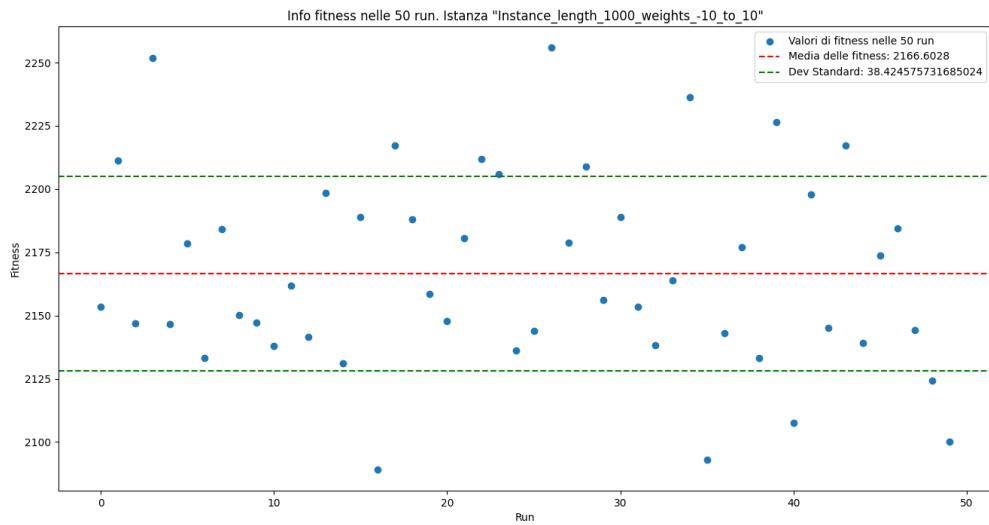


Figura 3.2: Scatterplot delle migliori soluzioni delle 50 run:

Istanza: 1000_-10_to_+10

Algoritmo: GA classico

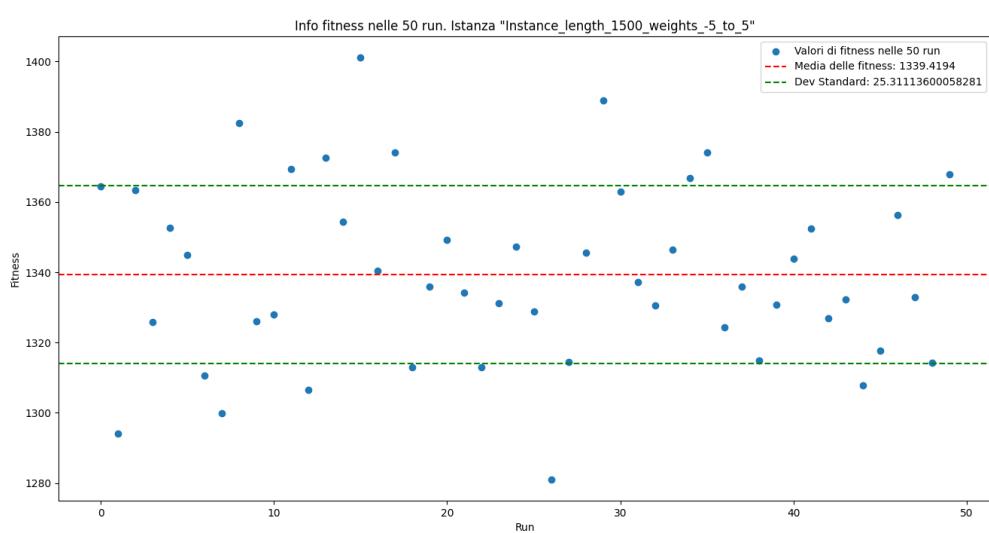


Figura 3.3: Scatterplot delle migliori soluzioni delle 50 run:

Istanza: 1500_-5_to_+5

Algoritmo: GA classico

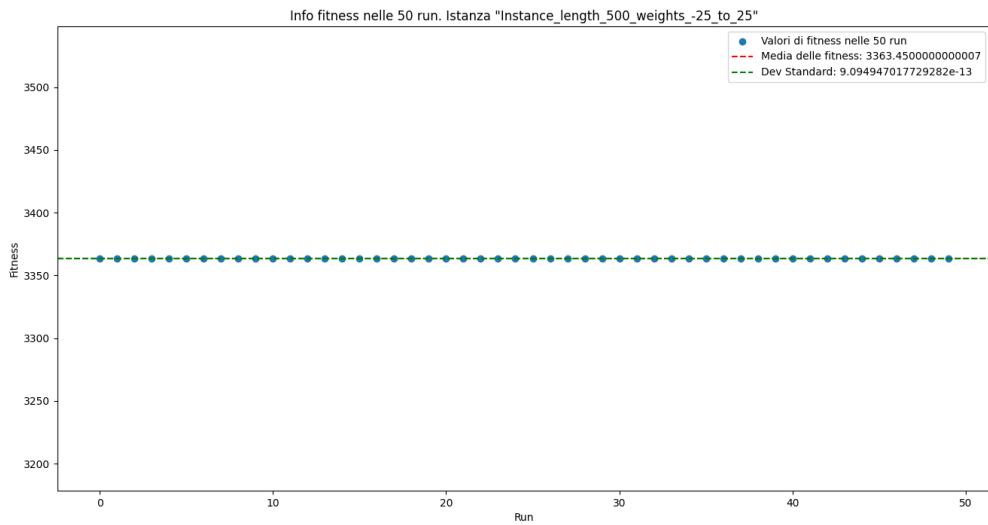


Figura 3.4: Scatterplot delle migliori soluzioni delle 50 run:

Istanza: 500_-25_to_+25

Algoritmo: SelectionGA

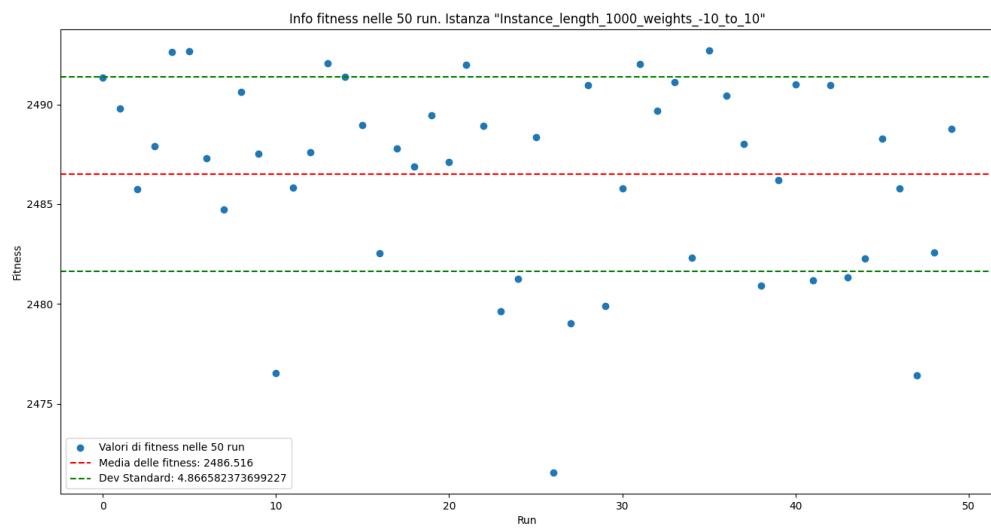


Figura 3.5: Scatterplot delle migliori soluzioni delle 50 run:

Istanza: 1000_-10_to_+10

Algoritmo: SelectionGA

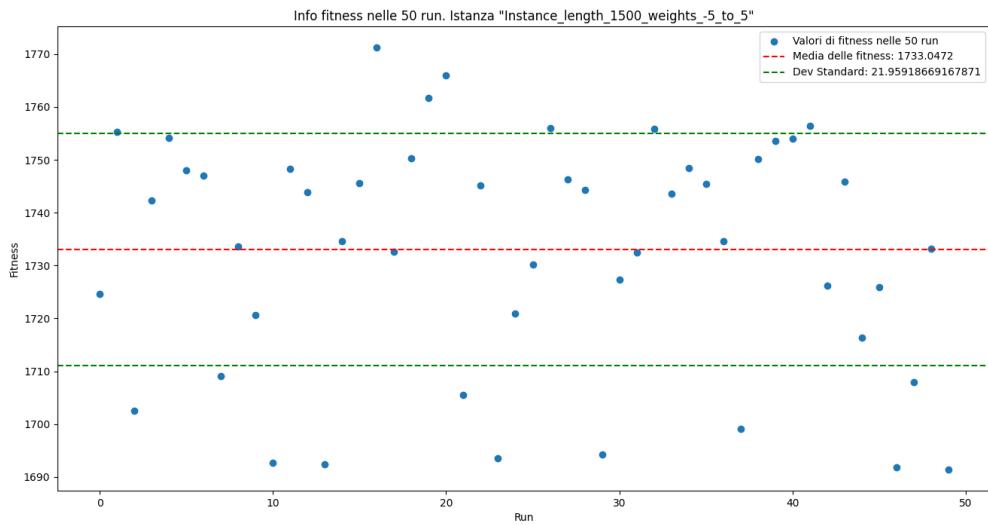


Figura 3.6: Scatterplot delle migliori soluzioni delle 50 run:

Istanza: 1500_-5_to_-5

Algoritmo: SelectionGA

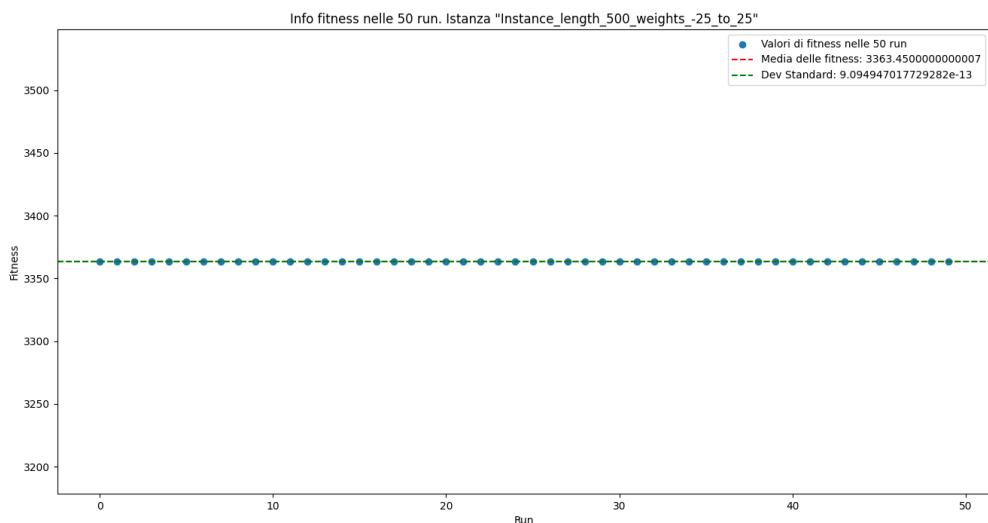


Figura 3.7: Scatterplot delle migliori soluzioni delle 50 run:

Istanza: 500_-25_to_+25

Algoritmo: RandomGA

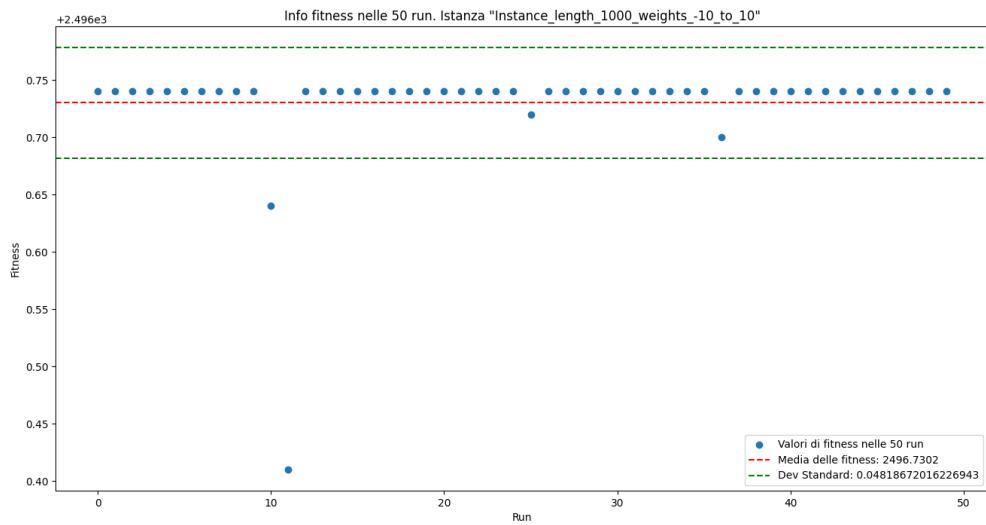


Figura 3.8: Scatterplot delle migliori soluzioni delle 50 run:

Istanza: 1000_-10_to_+10

Algoritmo: RandomGA

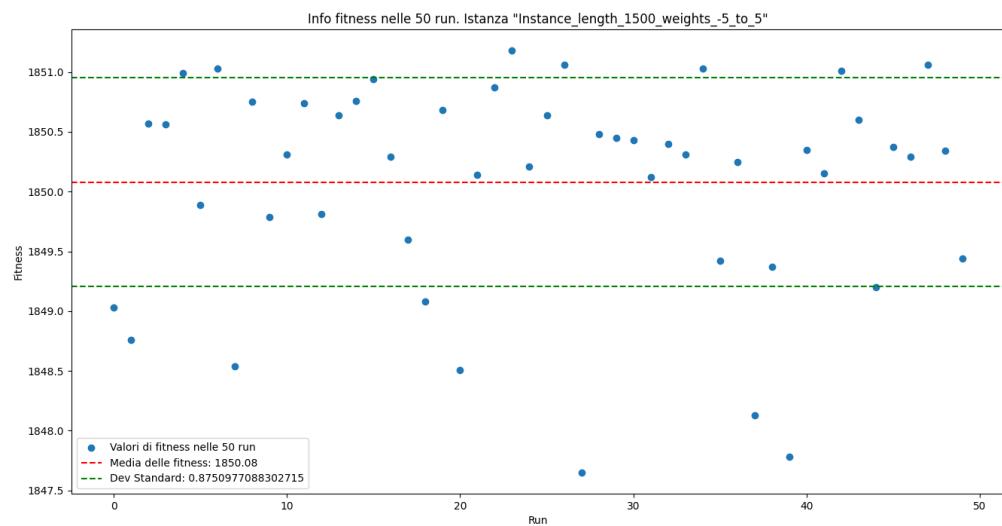


Figura 3.9: Scatterplot delle migliori soluzioni delle 50 run:

Istanza: 1500_-5_to_+5

Algoritmo: RandomGA

3.2 Convergenza

Sono di seguito riportati dei grafici riguardanti la convergenza dell'algoritmo nei diversi esperimenti, in alcune delle 50 run eseguite per ognuno di essi.

I grafici riportano le seguenti informazioni:

- **Andamento della fitness media nel corso della run** (curva blu)
- **Valore della migliore soluzione trovata nella run** (linea verde)
- **Valore della soluzione ottima per l'istanza considerata** (linea rossa)

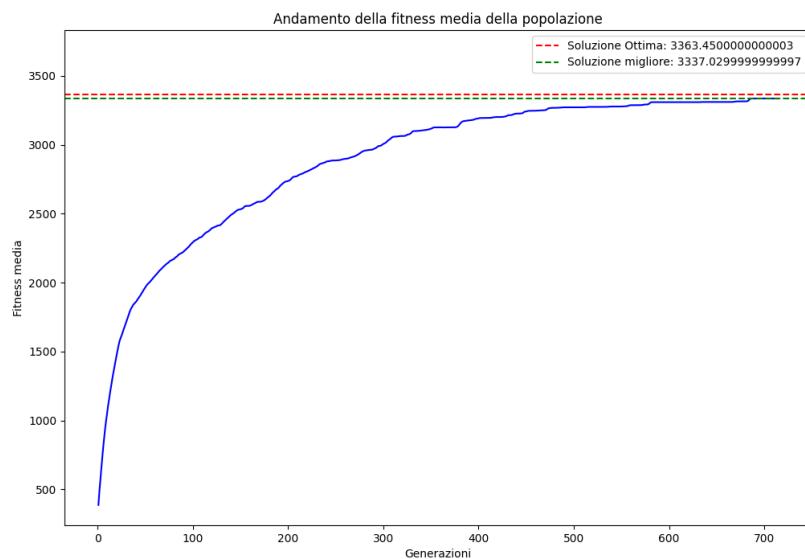


Figura 3.10: Convergenza dell'esperimento:

Istanza: 500_-25_to_+25

Algoritmo: GA classico

Run: 40/50

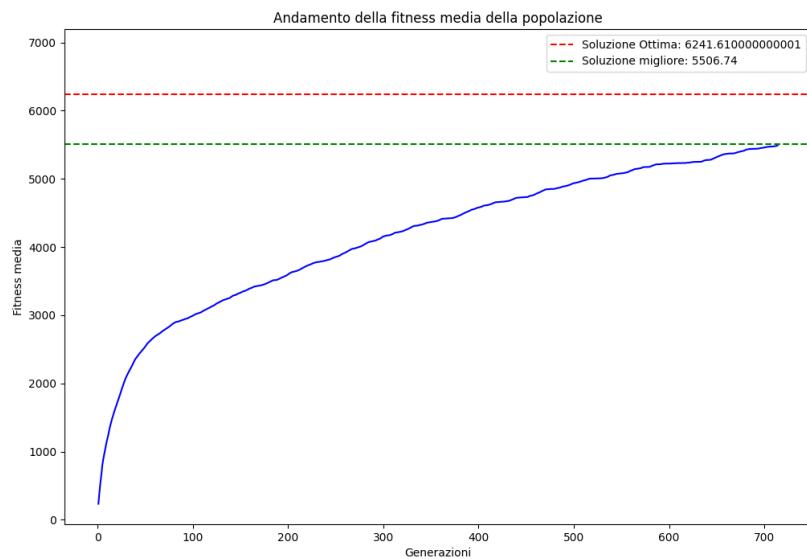


Figura 3.11: Convergenza dell'esperimento:

Istanza: 1000_-25_to_+25

Algoritmo: GA classico

Run: 28/50

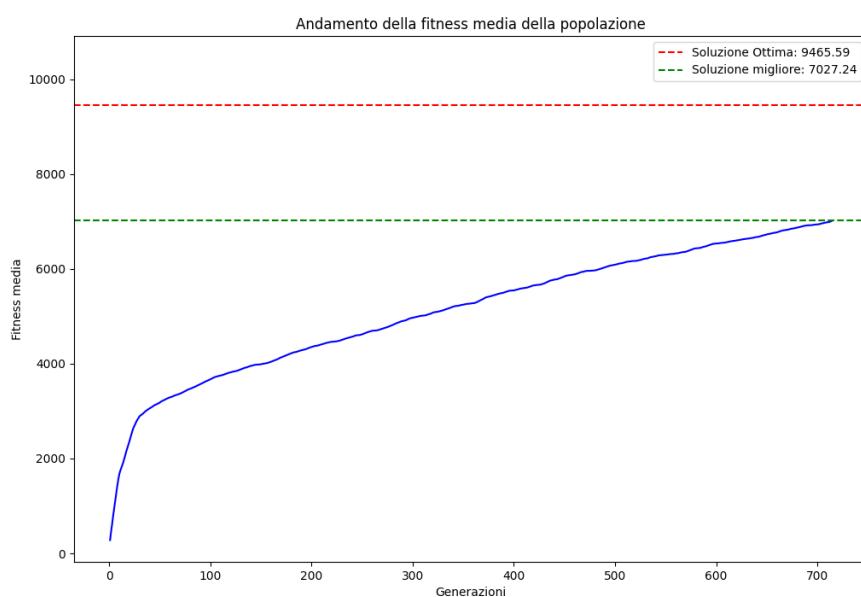


Figura 3.12: Convergenza dell'esperimento:

Istanza: 1500_-25_to_+25

Algoritmo: GA classico

Run: 21/50

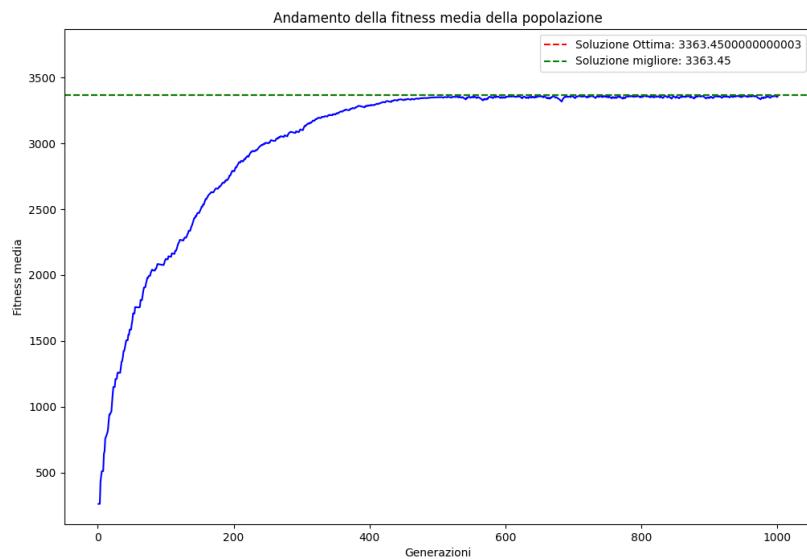


Figura 3.13: Convergenza dell'esperimento:

Istanza: 500_-25_to_+25

Algoritmo: SelectionGA

Run: 49/50

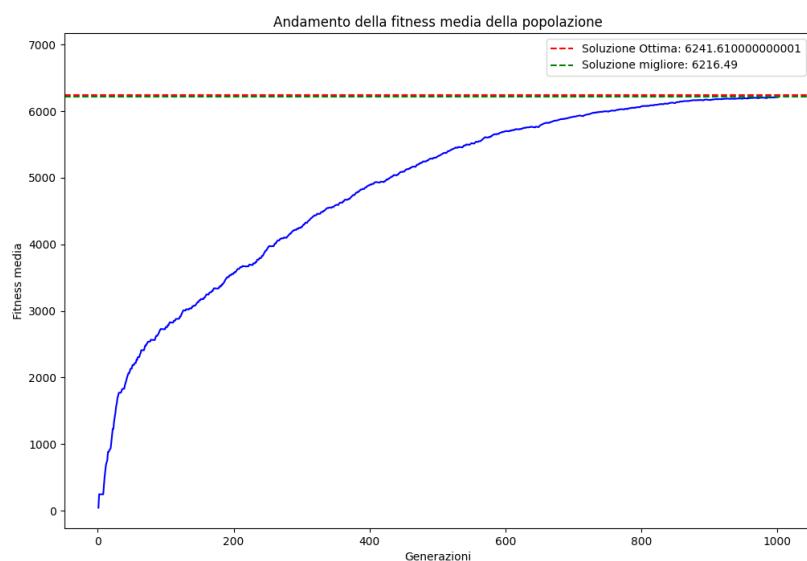


Figura 3.14: Convergenza dell'esperimento:

Istanza: 1000_-25_to_+25

Algoritmo: SelectionGA

Run: 23/50

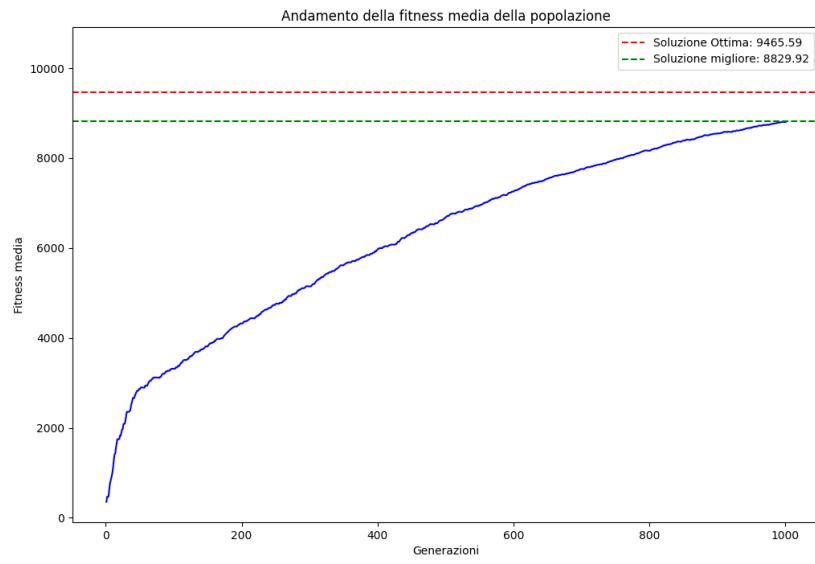


Figura 3.15: Convergenza dell'esperimento:
Istanza: 1500_-25_to_+25
Algoritmo: SelectionGA
Run: 6/50

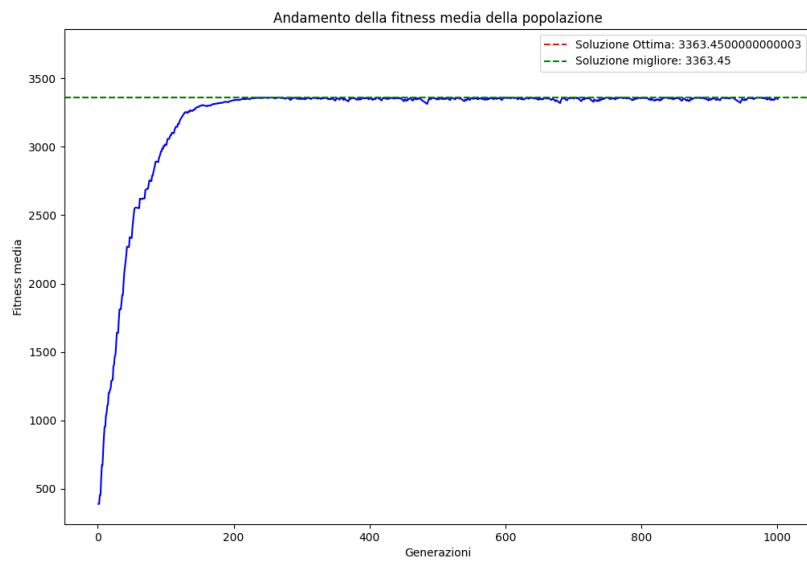


Figura 3.16: Convergenza dell'esperimento:
Istanza: 500_-25_to_+25
Algoritmo: RandomGA
Run: 14/50

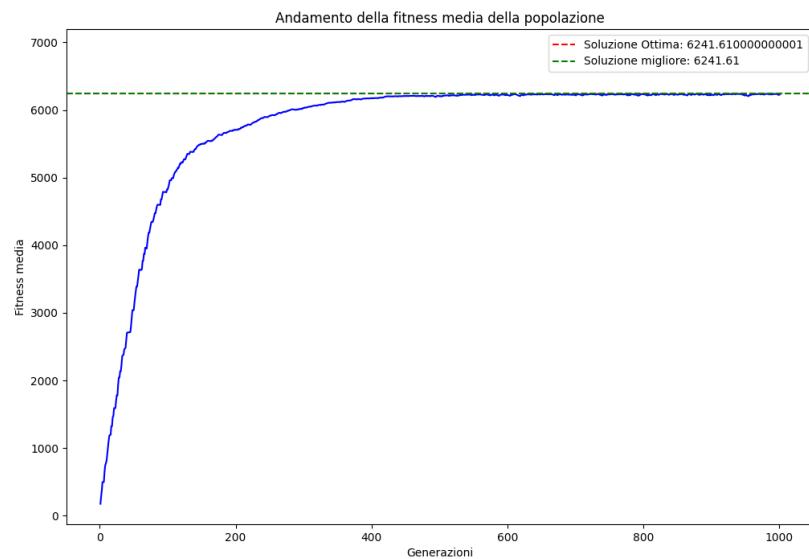


Figura 3.17: Convergenza dell'esperimento:

Istanza: 1000_-25_to_+25

Algoritmo: RandomGA

Run: 48/50

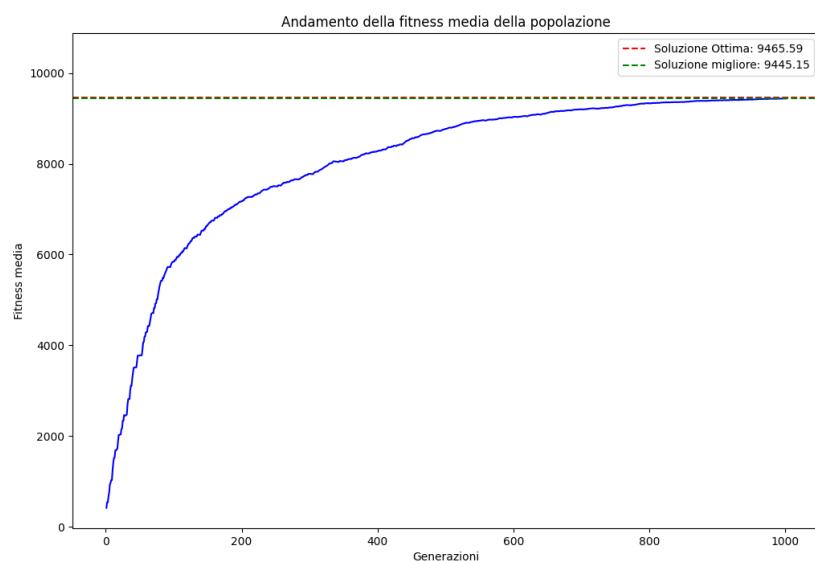


Figura 3.18: Convergenza dell'esperimento:

Istanza: 1500_-25_to_+25

Algoritmo: RandomGA

Run: 17/50

Dai precedenti grafici, si può notare quanto già espresso in parte nella sezione 3.1.1:

- **GA classico:**

- Nelle istanze con vettore dei pesi con lunghezza $L = 500$, l'algoritmo riesce a raggiungere una soluzione vicinissima a quella ottimale. La fitness cresce abbastanza rapidamente, e già a poco più di metà della run l'algoritmo trova soluzioni subottimali, raffinandole poi alle iterazioni successive.
- Nelle istanze con vettore dei pesi con lunghezza $L = 1000$, l'algoritmo riesce a raggiungere una soluzione non ottimale ma abbastanza buona, con un errore pari a circa il 10% tra la soluzione migliore trovata e quella ottima. Dal punto di vista della convergenza, possiamo quindi dire che l'algoritmo non trova la soluzione ottima, bensì una soluzione sub-ottimale.
- Nelle istanze con vettore dei pesi con lunghezza $L = 1500$, l'algoritmo non riesce ad ottenere delle buone soluzioni, e ciò è chiaro dal fatto che si ha un errore pari a circa il 25% tra la soluzione migliore trovata e quella ottima.
- Si noti che viene sempre raggiunto il numero massimo di calcoli della fitness consentito, in tutte le tipologie di istanze.

- **SelectionGA:**

- Nelle istanze con vettore dei pesi con lunghezza $L = 500$, l'algoritmo riesce in ogni run a raggiungere la soluzione ottimale. La fitness cresce abbastanza rapidamente, e già a circa un quarto della run l'algoritmo trova soluzioni subottimali, le quali vengono poi migliorate fino a raggiungere l'ottimo.
- Anche nel caso di istanze con vettore dei pesi con lunghezza $L = 1000$, l'algoritmo riesce a raggiungere la soluzione ottima o molto vicina ad essa. Dal punto di vista della convergenza, l'algoritmo raggiunge una soluzione sub-ottima a poco più di metà della run, fino a convergere alla soluzione ottima nelle ultime iterazioni.
- Nelle istanze con vettore dei pesi con lunghezza $L = 1500$, l'algoritmo riesce ad ottenere delle soluzioni sub-ottimali, con un errore pari a circa il 7% tra la soluzione migliore trovata e quella ottima. L'algoritmo converge a tali soluzioni sub-ottimali soltanto alla fine della run.

- **RandomGA:**

- Nelle istanze con vettore dei pesi con lunghezza $L = 500$, l'algoritmo riesce in ogni run a raggiungere la soluzione ottimale. La fitness cresce molto rapidamente, e già dopo poche generazioni l'algoritmo trova soluzioni subottimali, mentre già a partire da circa un quarto della run l'algoritmo riesce a convergere alla soluzione ottima.
- Anche nel caso di istanze con vettore dei pesi con lunghezza $L = 1000$, l'algoritmo riesce a raggiungere sempre la soluzione ottima. Dal punto di vista della convergenza, l'algoritmo raggiunge una soluzione sub-ottima a circa un quarto della run, fino a convergere alla soluzione ottima a circa metà dell'esecuzione.
- Nelle istanze con vettore dei pesi con lunghezza $L = 1500$, l'algoritmo riesce ad ottenere delle soluzioni con valore di fitness quasi uguale rispetto alla soluzione ottima, con un ridotto tasso di errore.

3.3 Variazione dei parametri dell'algoritmo

Sono di seguito riportate delle analisi riguardanti l'andamento dei valori dei parametri degli algoritmi SelectionGA e RandomGA, i cui valori vengono dinamicamente modificati grazie alla regressione lineare e all'approccio di Reinforcement Learning illustrati in precedenza.

I parametri propri degli algoritmi SelectionGA e RandomGA sono riportati nelle seguenti tabelle.

Simbolo	Parametro	Valore iniziale / default
k	#generazioni	1000
n	dimensione popolazione	350
v	dimensione buffer di regressione	40
p_c	probabilità di crossover	0.5
p_m	probabilità di mutazione	0.1
<u>$probs_crossover$</u>	probabilità di scelta dei diversi metodi di crossover (single-point, k-point, uniform)	[0.33,0.33,0.34]
<u>$probs_selection$</u>	probabilità di scelta dei diversi metodi di selezione (tournament, roulette, elitism, rank)	[0.25,0.25,0.25,0.25]
ϵ	goodness threshold	0.2
p_c^f	Crossover probability factor	0.7
p_c^l	Crossover probability level	20
p_m^f	Mutation probability factor	0.3
p_m^l	Mutation probability level	50
p_{max}^c	Probabilità massima di crossover	0.8
p_{min}^c	Probabilità minima di crossover	0.5
p_{max}^m	Probabilità massima di mutazione	0.3
p_{min}^m	Probabilità minima di mutazione	0.1
T_{max}	# calcoli fitness massimi	10^6
Δ	# Reward per Reinforcement Learning	0.4

Tabella 3.10: Parametri degli algoritmi SelectionGA e RandomGA.

I parametri sottolineati sono quelli che vengono modificati dinamicamente durante l'esecuzione dell'algoritmo grazie all'applicazione della regressione lineare e del reinforcement learning.

3.3.1 Variazione dei parametri di crossover e mutazione

Sono riportati di seguito dei grafici che mostrano la variazione dei parametri di crossover e mutazione durante l'esecuzione dell'algoritmo (solo per alcune istanze, a titolo di esempio). Come già detto in precedenza, il valore è modificato applicando la regressione lineare, considerando le precedenti v iterazioni al fine di predire un buon valore per tali parametri.

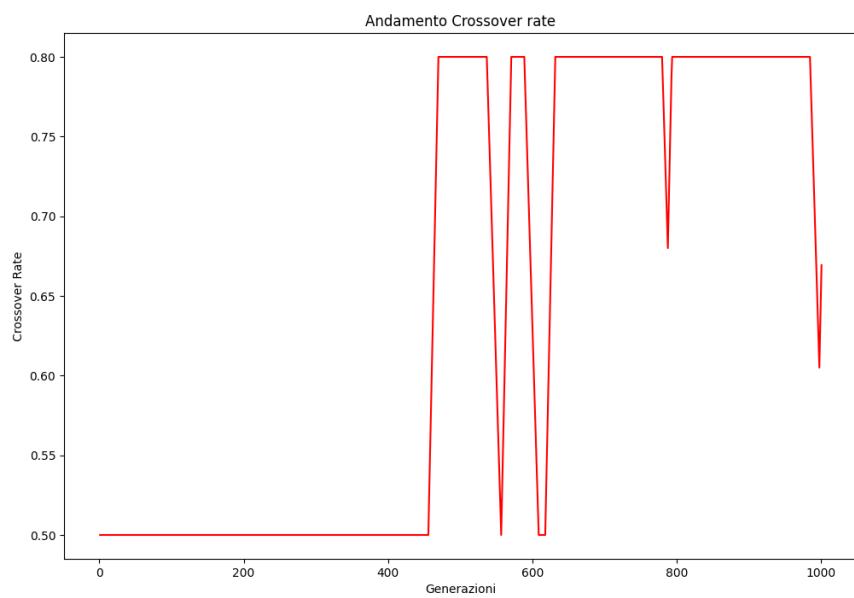


Figura 3.19: Variazione del crossover rate:

Istanza: 500_-10_to_+10

Algoritmo: SelectionGA

Run: 25/50

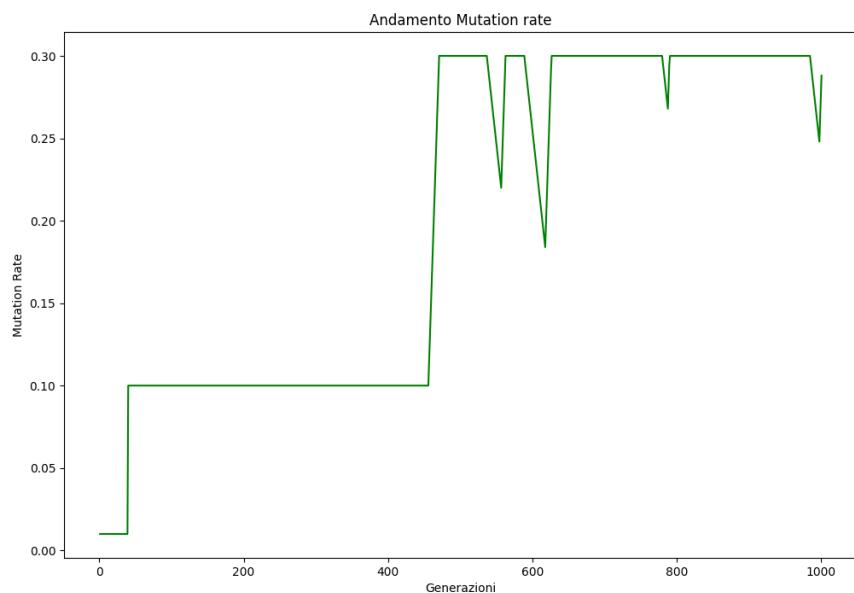


Figura 3.20: Variazione del mutation rate:

Istanza: 500_-10_to_+10

Algoritmo: SelectionGA

Run: 25/50

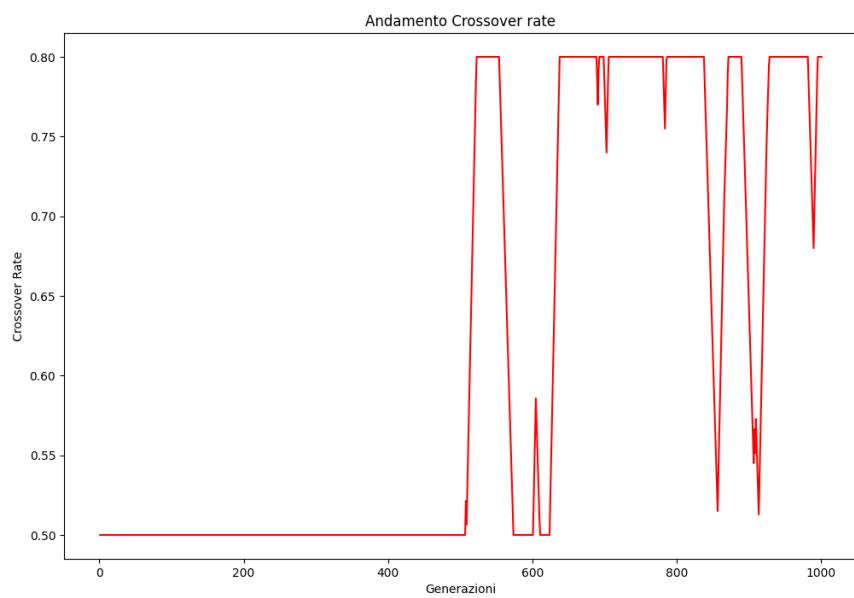


Figura 3.21: Variazione del crossover rate:

Istanza: 500_-25_to_+25

Algoritmo: SelectionGA

Run: 22/50

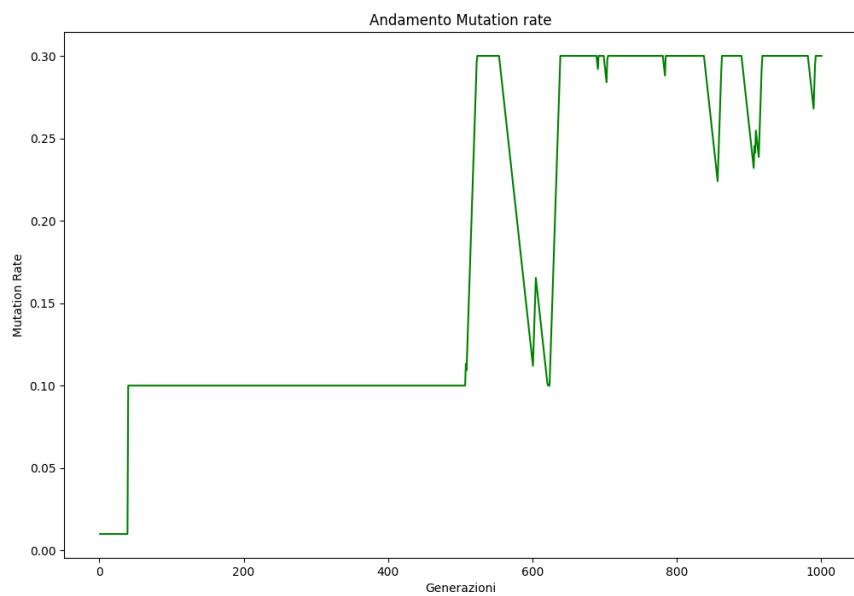


Figura 3.22: Variazione del mutation rate:

Istanza: 500_-25_to_+25

Algoritmo: SelectionGA

Run: 22/50

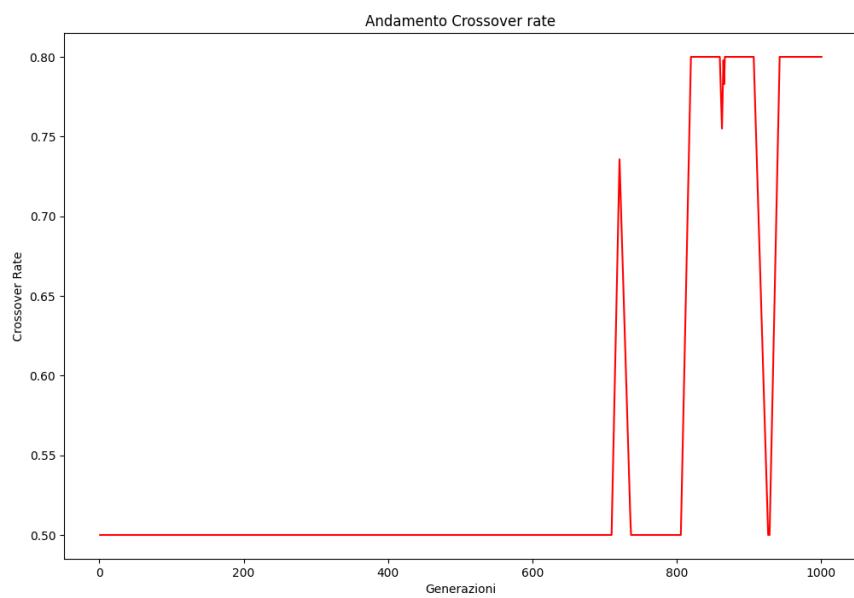


Figura 3.23: Variazione del crossover rate:

Istanza: 1000_-5_to_+5

Algoritmo: SelectionGA

Run: 49/50

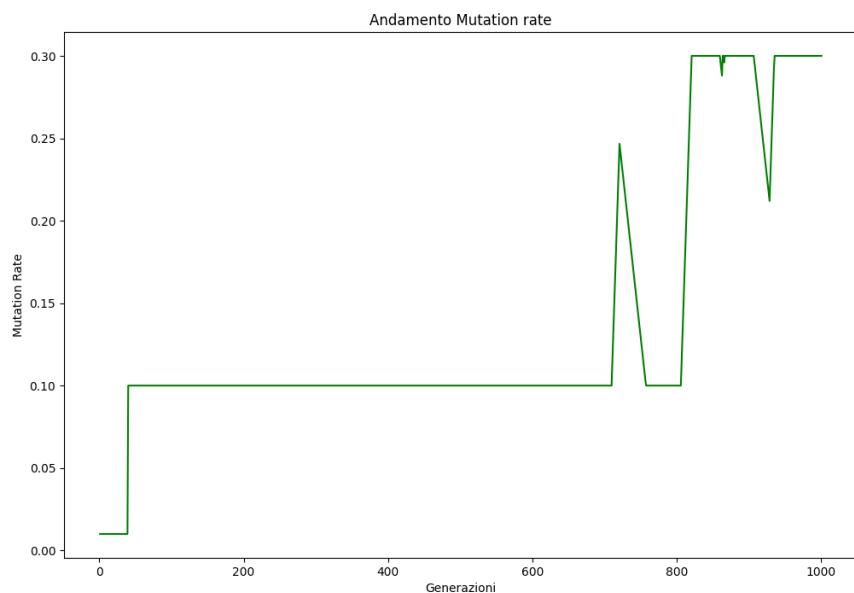


Figura 3.24: Variazione del mutation rate:

Istanza: 1000_-5_to_+5

Algoritmo: SelectionGA

Run: 49/50

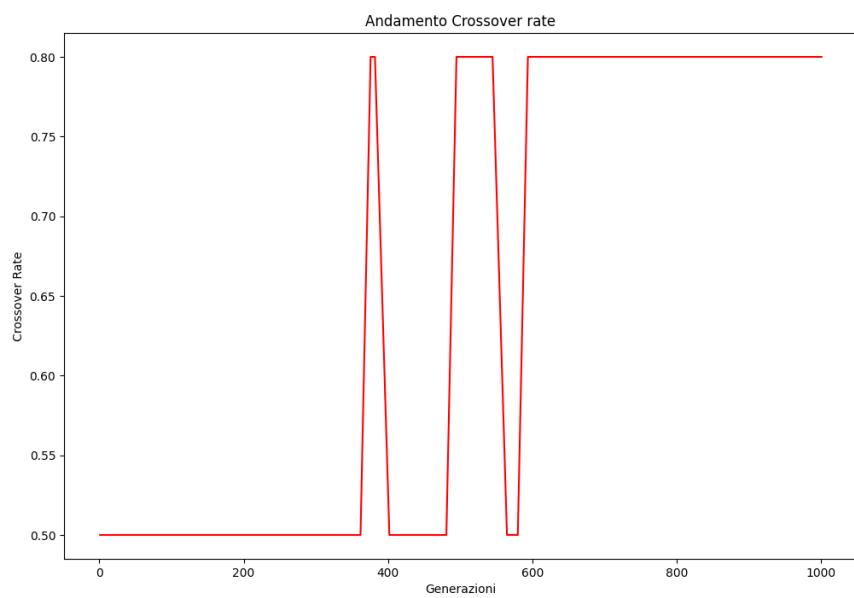


Figura 3.25: Variazione del crossover rate:

Istanza: 1000_-5_to_+5

Algoritmo: RandomGA

Run: 43/50

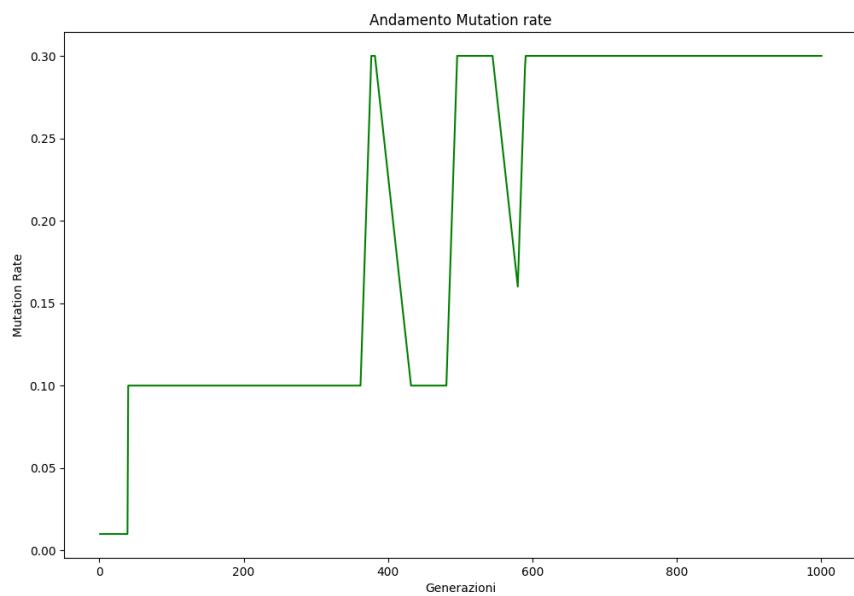


Figura 3.26: Variazione del mutation rate:

Istanza: 1000_-5_to_+5

Algoritmo: RandomGA

Run: 43/50

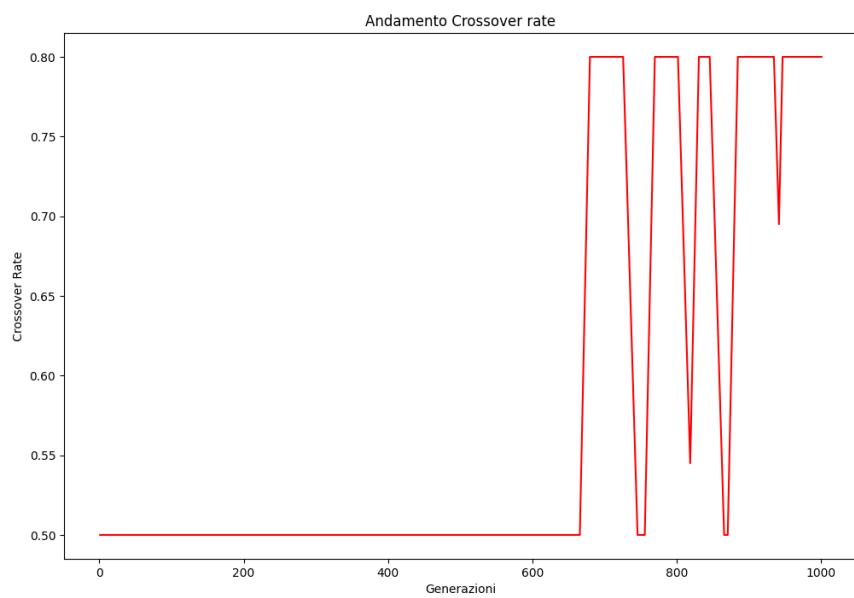


Figura 3.27: Variazione del crossover rate:

Istanza: 1500_-10_to_+10

Algoritmo: RandomGA

Run: 4/50

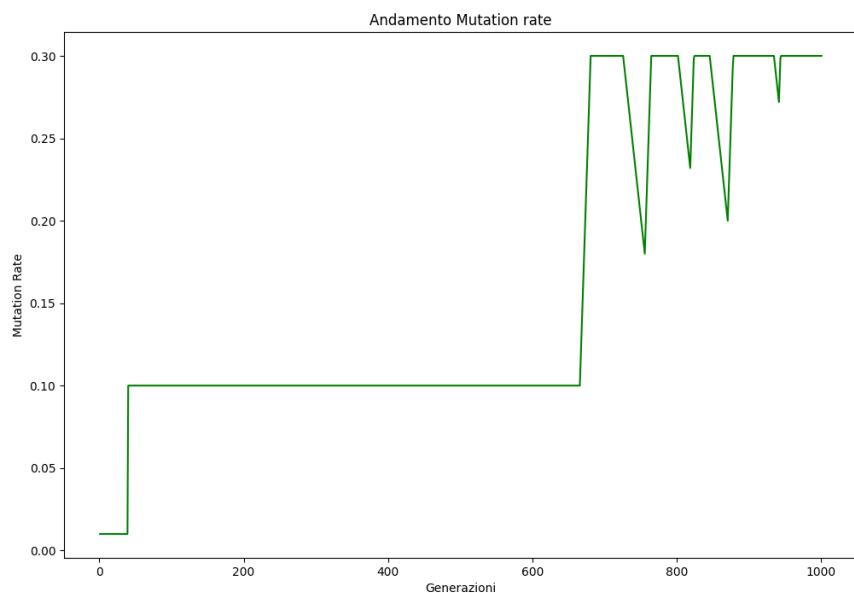


Figura 3.28: Variazione del mutation rate:

Istanza: 1500_-10_to_+10

Algoritmo: RandomGA

Run: 4/50

Dai grafici relativi ai parametri di crossover e mutazione, si può notare come nel corso dell'esecuzione dell'algoritmo tali parametri vengano modificati dinamicamente.

In particolare, da questi pochi esempi si può notare che:

- Per quanto riguarda il crossover rate:
 - **Inizialmente** tende a rimanere stabile, continuando ad assumere il valore di default
 - **A circa metà della run** il suo valore viene modificato
- Per quanto riguarda il mutation rate, possono essere fatte le stesse considerazioni. Si noti che **nelle prime v generazioni il mutation rate viene posto di default a 0.01 per evitare che le soluzioni prodotte inizialmente varino troppo**, a causa di un mutation rate troppo elevato. Quest'ultimo, dopo le prime v generazioni, viene poi modificato dinamicamente (con valore minimo di default pari a 0.1, come riportato in tabella 3.10)

L'andamento dei valori dei due parametri ci suggerisce in particolare che:

- Inizialmente, la fitness della popolazione cresce abbastanza rapidamente, e per questo motivo non si ha la necessità di modificare sin da subito i parametri di cross-over e mutazione
- Andando avanti nell'esecuzione dell'algoritmo, possono verificarsi tre casi:
 - **La fitness continua a crescere**
 - **La fitness diminuisce**
 - **La fitness rimane pressochè stabile**

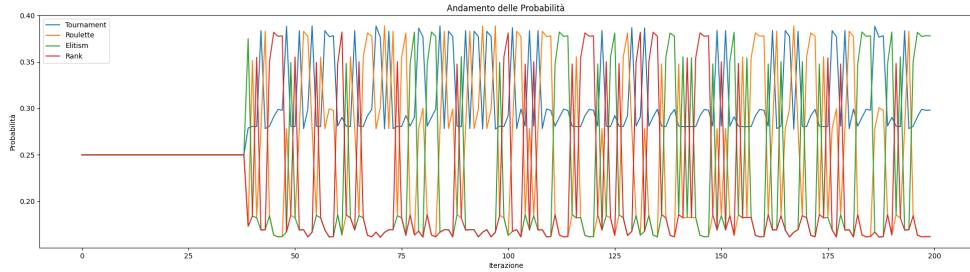
Negli ultimi due casi, significa che l'algoritmo non sta riuscendo a migliorare le soluzioni trovate (ad esempio perchè è rimasto bloccato su un ottimo locale), ed è quindi necessario aiutare l'algoritmo a migliorare le soluzioni trovate, e ciò viene fatto proprio modificando dinamicamente i parametri di crossover e mutazione, al fine di "variare" maggiormente le soluzioni trovate.

3.3.2 Variazione delle probabilità di scelta dei metodi di selezione

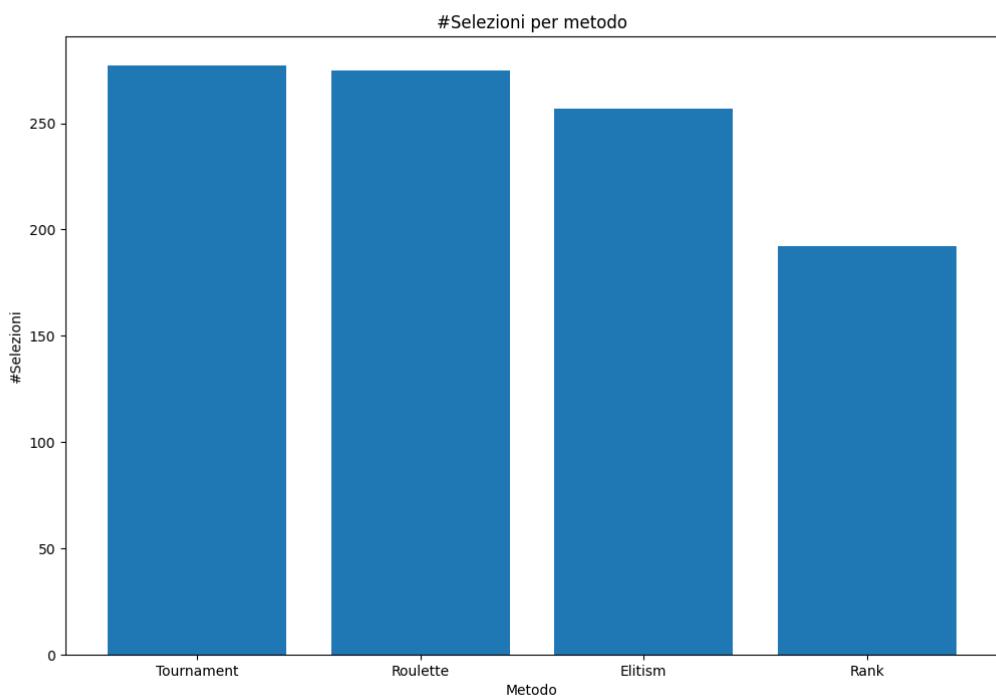
Sono di seguito riportate delle analisi riguardanti l'andamento della scelta dinamica dei metodi di selezione degli individui che daranno vita alla generazione successiva, ad ogni iterazione dell'algoritmo genetico.

In particolare, sono illustrati gli andamenti dei metodi di selezione riguardo ad un sottoinsieme degli esperimenti effettuati, sia per l'algoritmo SelectionGA che per l'algoritmo RandomGA. Per ognuno di questi ultimi sono riportati:

- **Grafico dell'andamento delle probabilità (solamente prime 200 iterazioni, per questioni di spazio)**
- **Barplot che indica quante volte è stato scelto ognuno dei metodi**



(a) Andamento delle probabilità



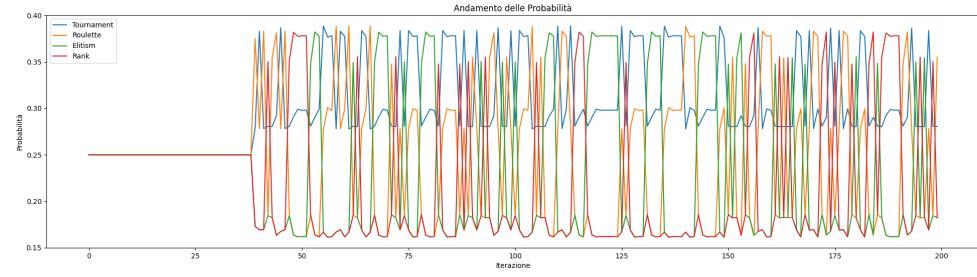
(b) # scelte per metodo

Figura 3.29: Scelta dei metodi di selezione:

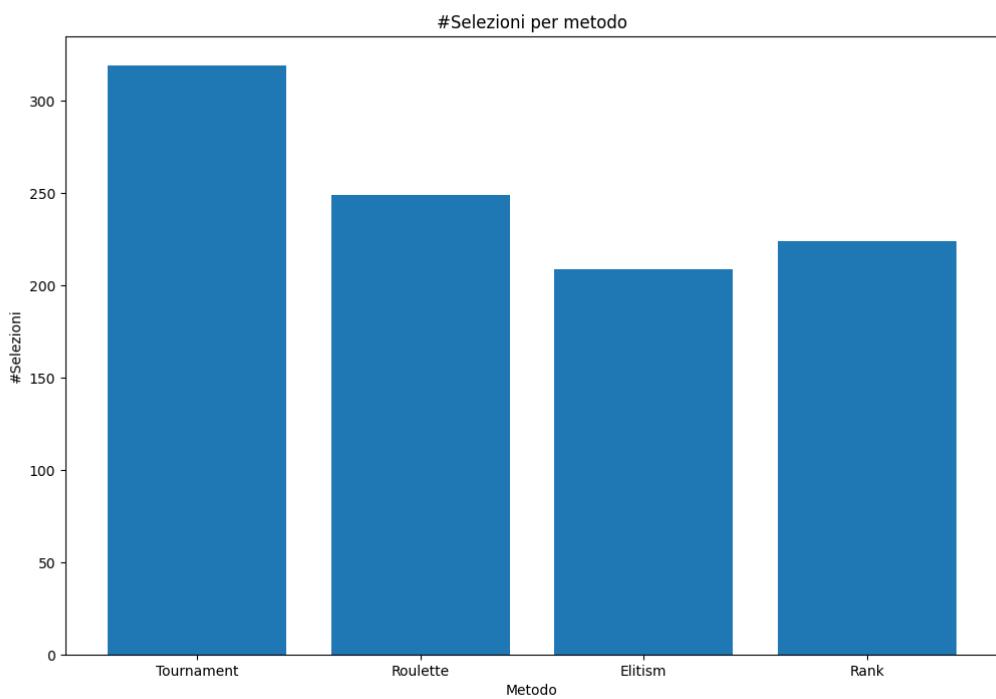
Istanza: 500_-10_to_+10

Algoritmo: SelectionGA

Run: 35/50



(a) Andamento delle probabilità



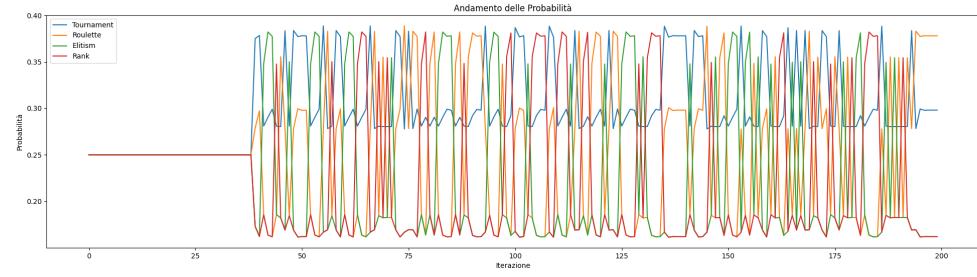
(b) # scelte per metodo

Figura 3.30: Scelta dei metodi di selezione:

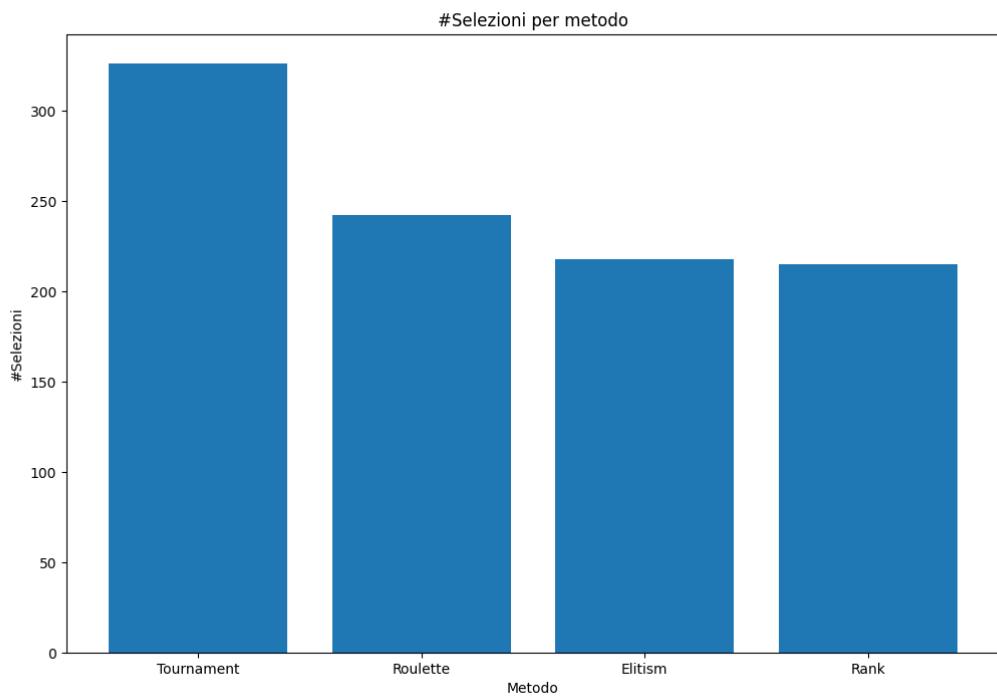
Istanza: 1000_-25_to_+25

Algoritmo: SelectionGA

Run: 6/50



(a) Andamento delle probabilità



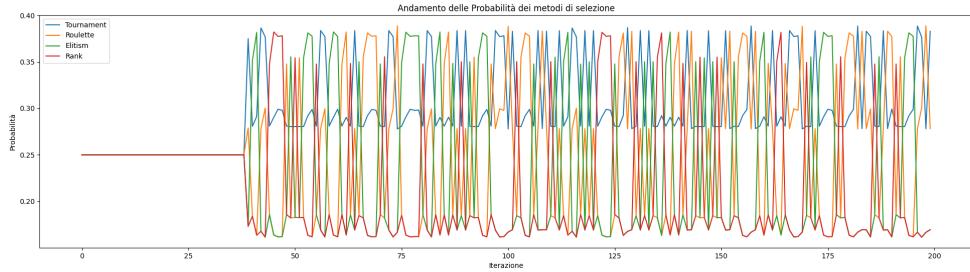
(b) # scelte per metodo

Figura 3.31: Scelta dei metodi di selezione:

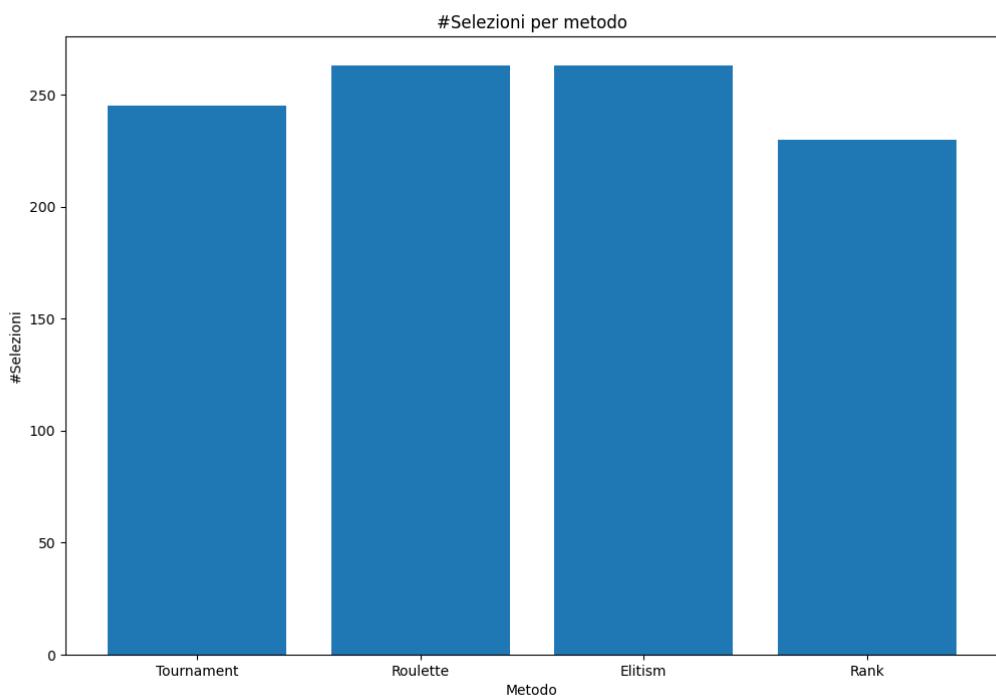
Istanza: 1500_-5_to_+5

Algoritmo: SelectionGA

Run: 19/50



(a) Andamento delle probabilità



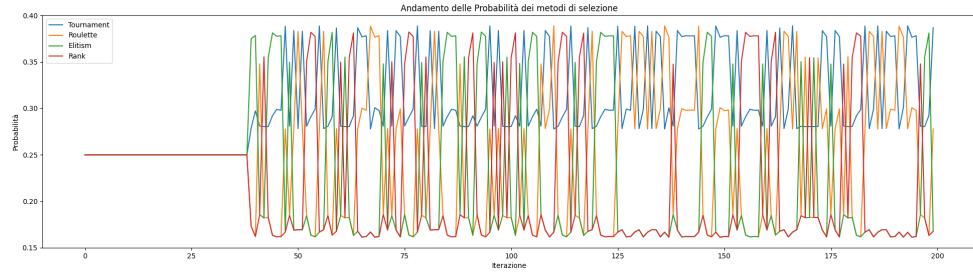
(b) # scelte per metodo

Figura 3.32: Scelta dei metodi di selezione:

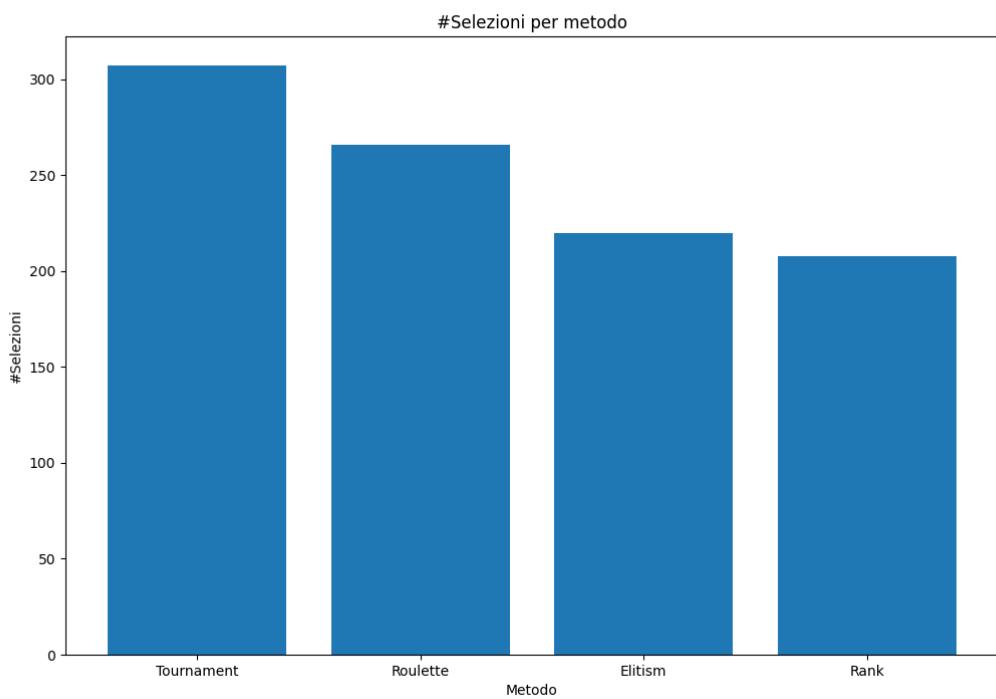
Istanza: 500_-10_to_+10

Algoritmo: RandomGA

Run: 11/50



(a) Andamento delle probabilità



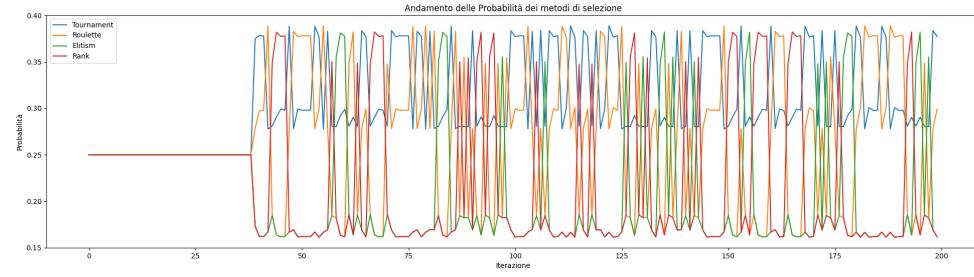
(b) # scelte per metodo

Figura 3.33: Scelta dei metodi di selezione:

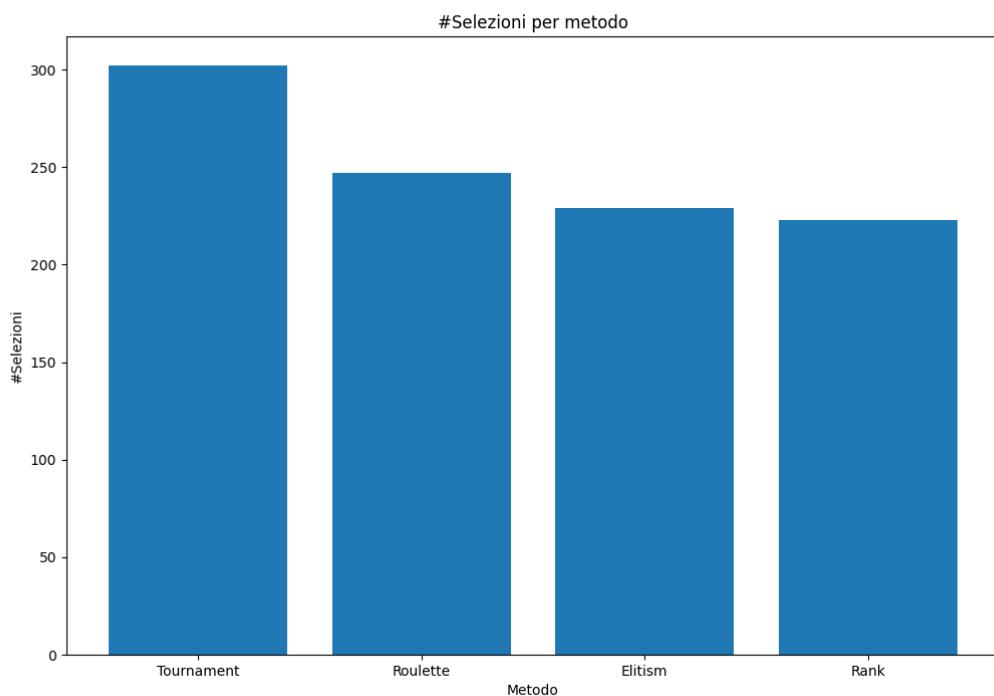
Istanza: 1000_-25_to_+25

Algoritmo: RandomGA

Run: 30/50



(a) Andamento delle probabilità



(b) # scelte per metodo

Figura 3.34: Scelta dei metodi di selezione:

Istanza: 1500_-5_to_+5

Algoritmo: RandomGA

Run: 17/50

3.3.3 Variazione delle probabilità di scelta dei metodi di selezione (interi esperimenti)

I grafici precedenti mostravano l'andamento della scelta dei diversi metodi di selezione nelle singole run. Poichè però i dati delle singole run potrebbero essere fortemente influenzati dalla stocasticità propria dell'algoritmo genetico, sono di seguito riportati anche dei grafici che mostrano quante volte un singolo metodo di selezione è stato scelto nel totale delle 50 run per le singole istanze.

A titolo di esempio, si riportano esempi relativi solamente a 3 istanze, sia per SelectionGA che per RandomGA.

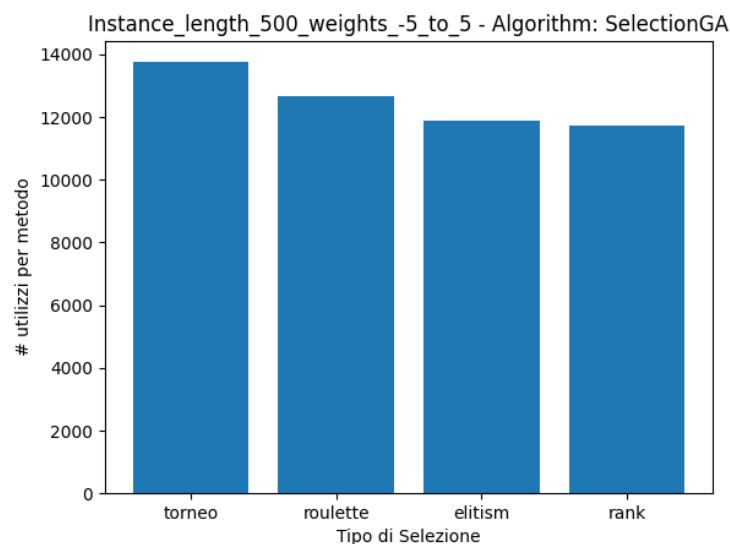


Figura 3.35: Scelta dei metodi di selezione (intera run):

Istanza: 500_-5_to_+5

Algoritmo: SelectionGA

Valori esatti:

Tournament	Roulette	Elitism	Rank
13706	12679	11902	11713

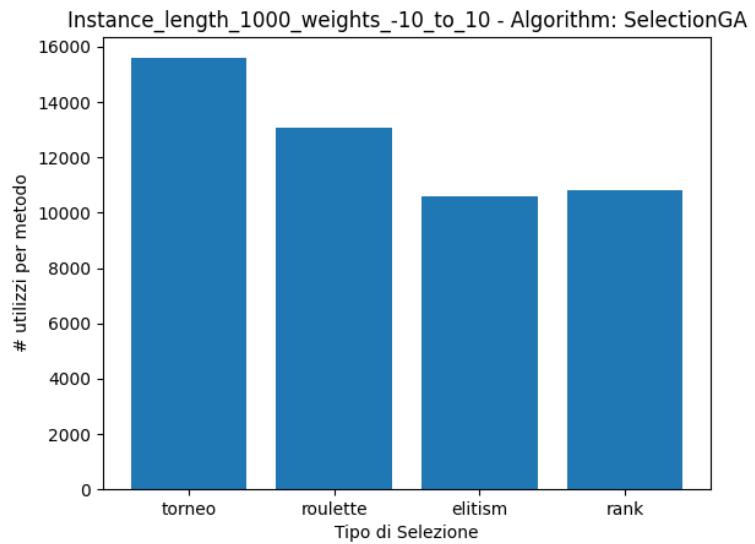


Figura 3.36: Scelta dei metodi di selezione (intera run):

Istanza: 1000_-10_to_+10

Algoritmo: SelectionGA

Valori esatti:

Tournament	Roulette	Elitism	Rank
15589	13018	10595	10798

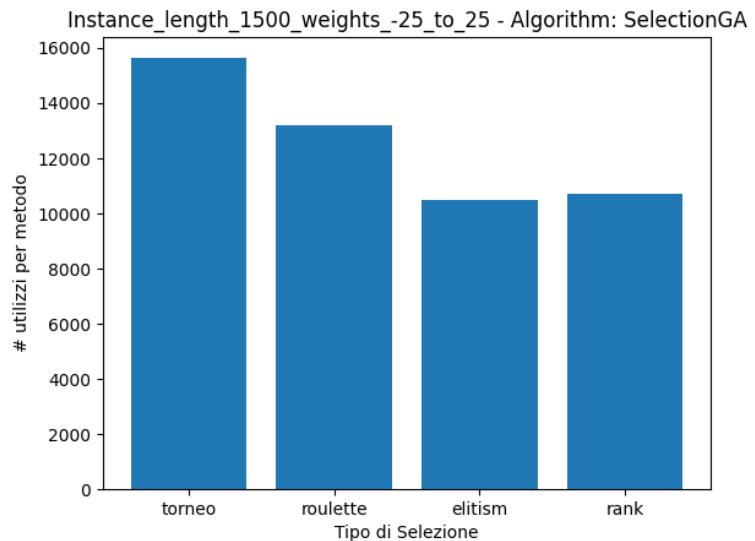


Figura 3.37: Scelta dei metodi di selezione (intera run):

Istanza: 1500_-25_to_+25

Algoritmo: SelectionGA

Tournament	Roulette	Elitism	Rank
15634	13147	10486	10733

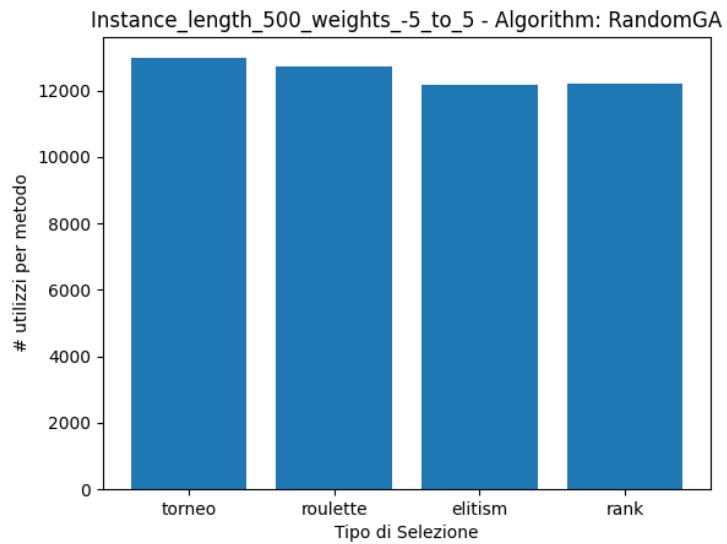


Figura 3.38: Scelta dei metodi di selezione (intera run):

Istanza: 500_-5_to_+5

Algoritmo: RandomGA

Tournament	Roulette	Elitism	Rank
12921	12715	12158	12206

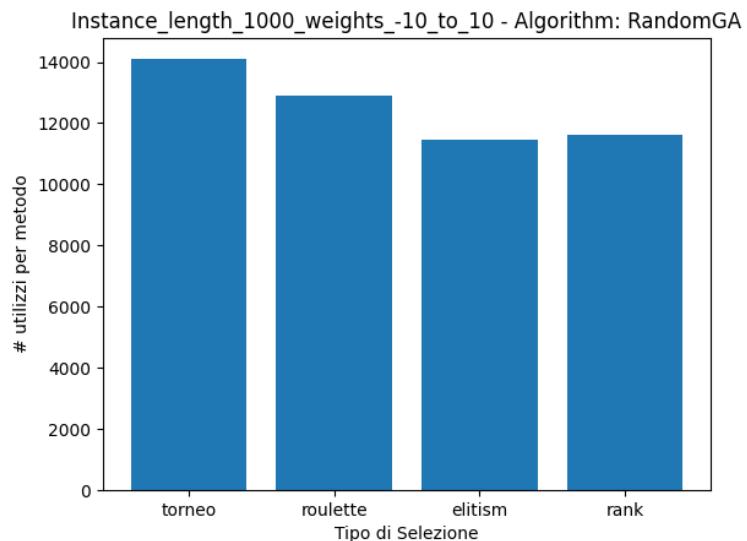


Figura 3.39: Scelta dei metodi di selezione (intera run):

Istanza: 1000_-10_to_+10

Algoritmo: RandomGA

Tournament	Roulette	Elitism	Rank
13349	12440	12101	12110

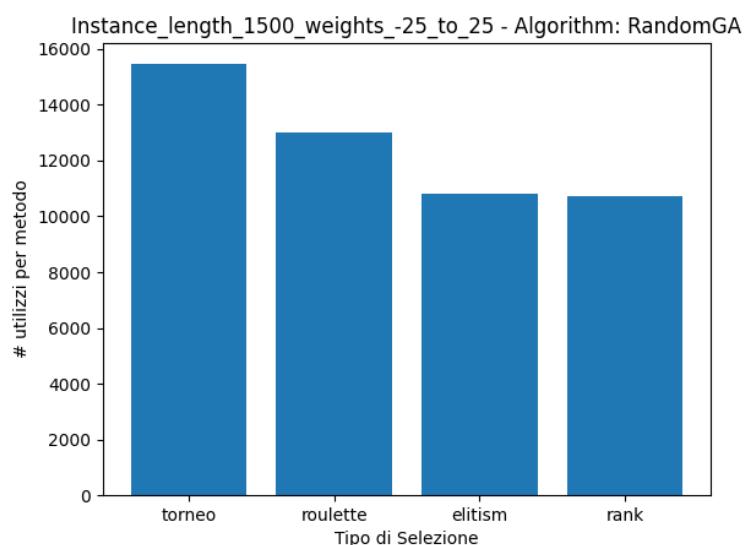


Figura 3.40: Scelta dei metodi di selezione (intera run):

Istanza: 1500_-25_to_+25

Algoritmo: RandomGA

Tournament	Roulette	Elitism	Rank
15407	13020	10833	10740

Come si può notare dai grafici precedenti, si evince che i metodi di selezione maggiormente usati dall'algoritmo sono:

- **Tournament selection**
- **Roulette selection**

In particolare, la **Roulette Selection** sembra quella che ottiene risultati migliori tra tutti i metodi di selezione, mentre al contrario quella che sembra ottenere i risultati peggiori sono la Rank Selection e l'Elitism Selection.

Ciò è evidente anche osservando i grafici relativi all'andamento delle probabilità dei diversi metodi:

- le curva relativa a tournament selection (curva blu) è quella che "si mantiene" su valori di probabilità elevati per la maggior parte del tempo di esecuzione dell'algoritmo.
- la curva relativa alla Rank Selection (curva rossa) si mantiene invece su valori di probabilità molto bassi, ovvero non viene spesso scelto come metodo di selezione, rispetto agli altri metodi

Si presume quindi dai risultati ottenuti che questi metodi di selezione siano quelli che facciano crescere maggiormente la qualità delle soluzioni trovate dall'algoritmo.

E' importante notare infine che, nonostante i metodi migliori siano la Tournament Selection e la Roulette Selection, l'approccio probabilistico usato per la scelta dei metodi di selezione evita che l'algoritmo converga alla scelta di un singolo metodo di selezione o, al contrario, vi sia un metodo di selezione che non viene mai scelto.

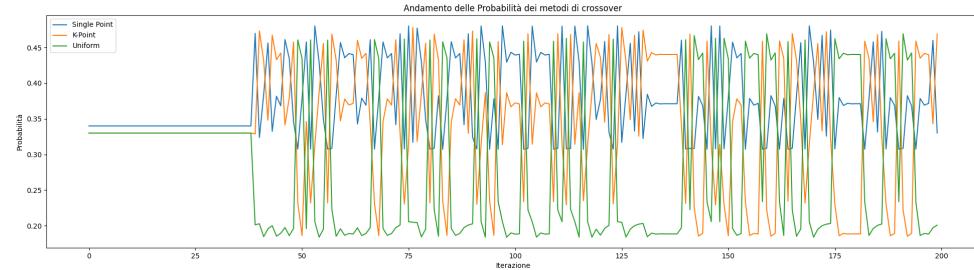
3.3.4 Variazione delle probabilità di scelta dei metodi di crossover (RandomGA)

Come già spiegato in precedenza, l'algoritmo RandomGA prevede che, oltre a diversi metodi di selezione (come in SelectionGA), sia possibile scegliere tra diversi metodi di cross-over in ognuna delle iterazioni. Sono di seguito riportate delle analisi riguardanti l'andamento della scelta dinamica dei metodi di crossover.

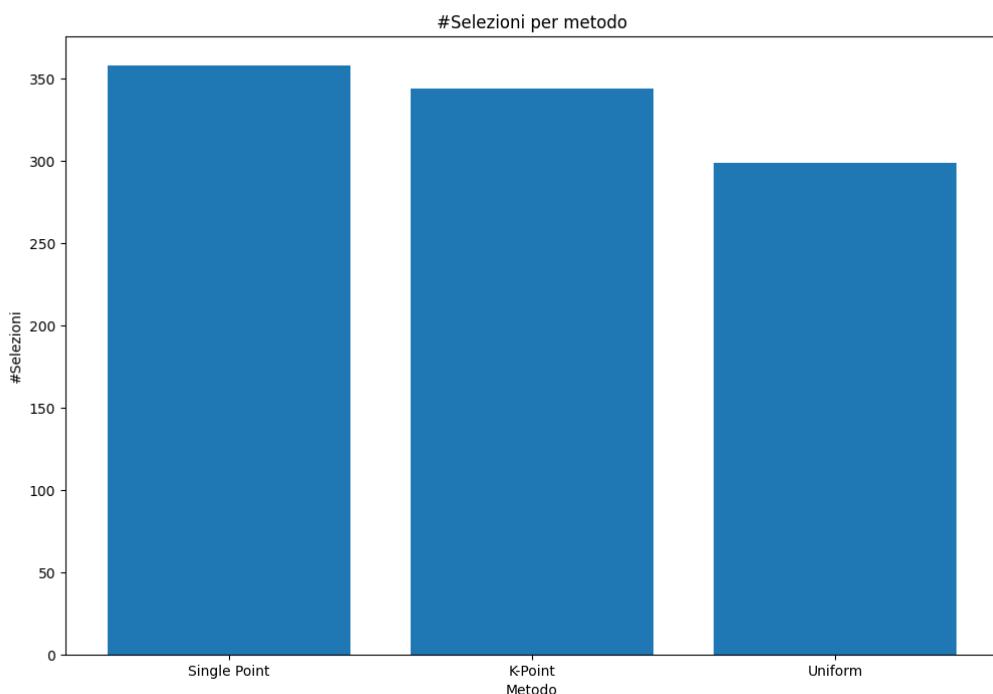
Come fatto nel caso dei metodi di selezione, sono nello specifico mostrati gli andamenti dei metodi di crossover solamente per un sottoinsieme degli esperimenti effettuati

Anche in questo caso, per ognuno di questi ultimi sono riportati:

- **Grafico dell'andamento delle probabilità (solamente prime 200 iterazioni, per questioni di spazio)**
- **Barplot che indica quante volte è stato scelto ognuno dei metodi**



(a) Andamento delle probabilità



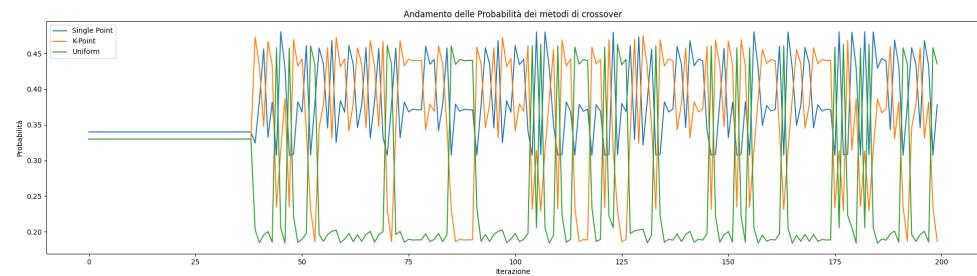
(b) # scelte per metodo

Figura 3.41: Scelta dei metodi di crossover:

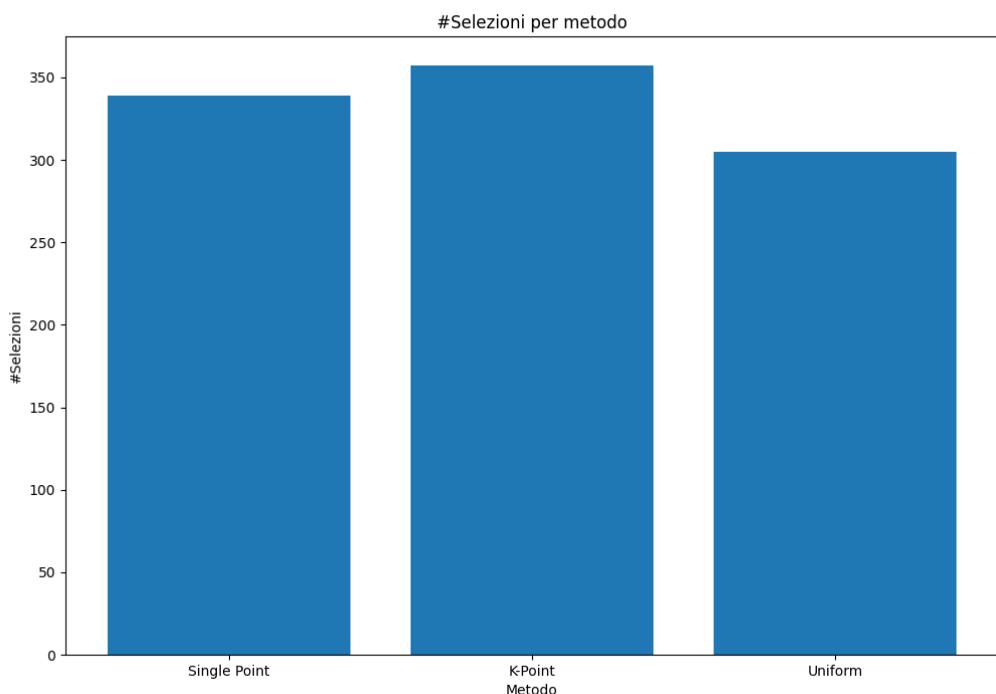
Istanza: 500_-10_to_+10

Algoritmo: RandomGA

Run: 33/50



(a) Andamento delle probabilità



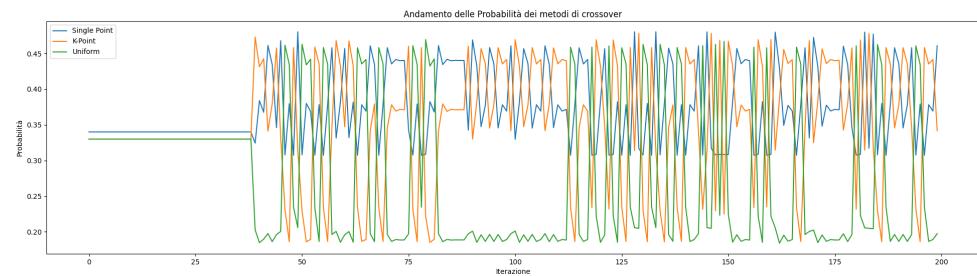
(b) # scelte per metodo

Figura 3.42: Scelta dei metodi di crossover:

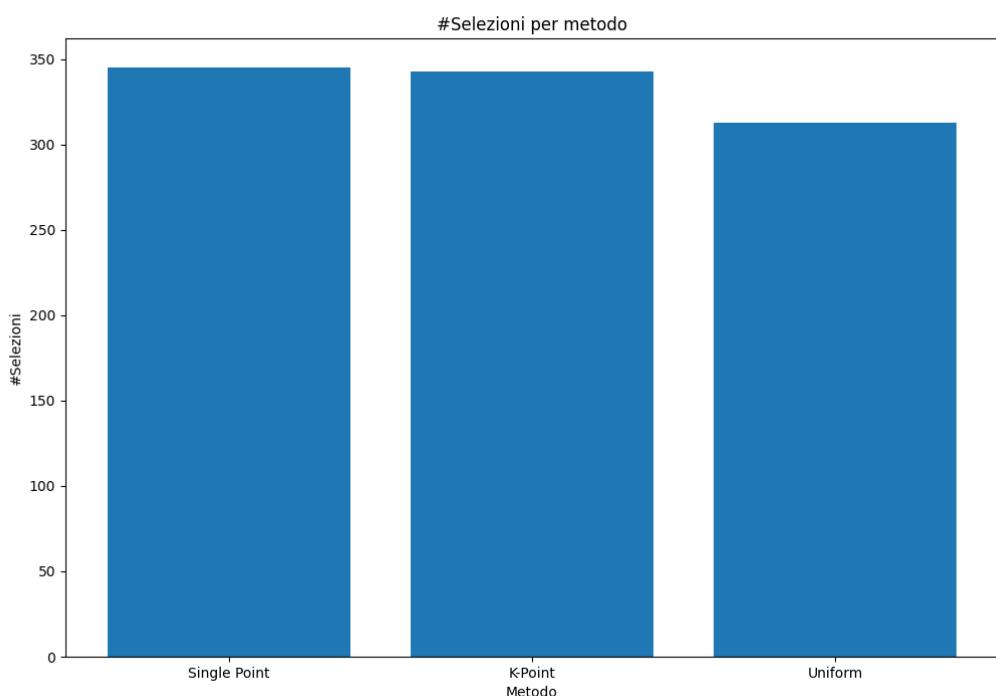
Istanza: 1000_-5_to_+5

Algoritmo: RandomGA

Run: 5/50



(a) Andamento delle probabilità



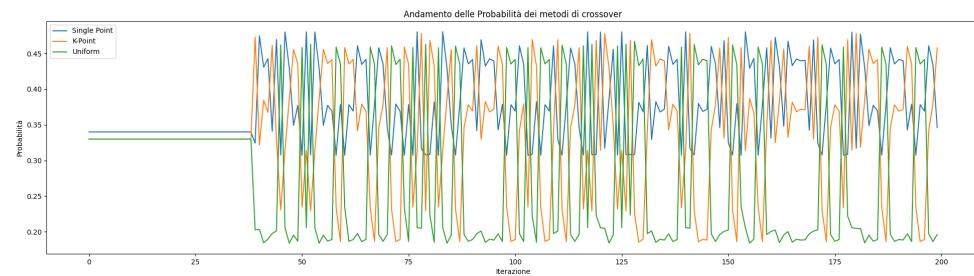
(b) # scelte per metodo

Figura 3.43: Scelta dei metodi di crossover:

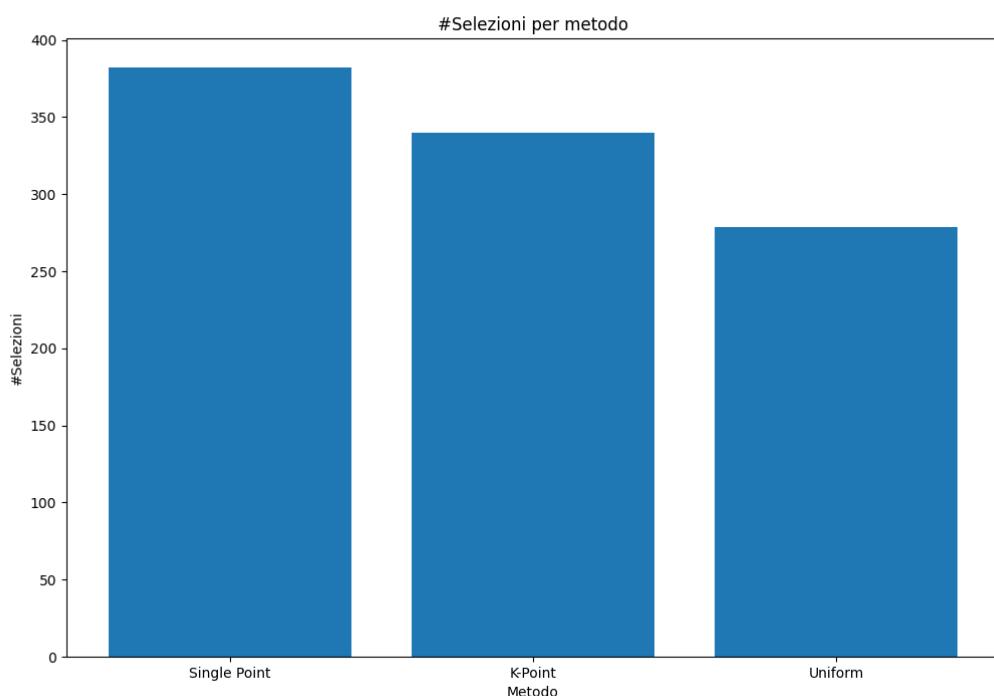
Istanza: 500_-25_to_+25

Algoritmo: RandomGA

Run: 32/50



(a) Andamento delle probabilità



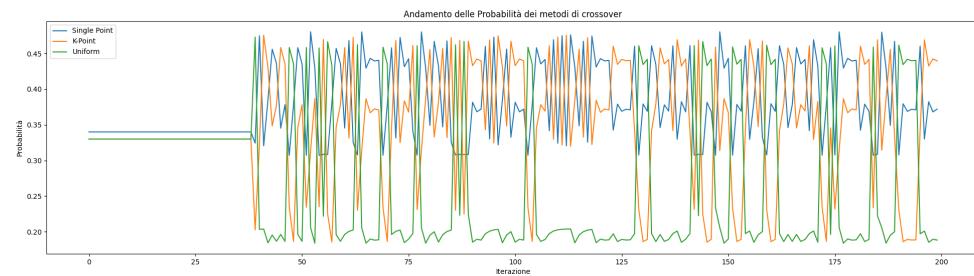
(b) # scelte per metodo

Figura 3.44: Scelta dei metodi di crossover:

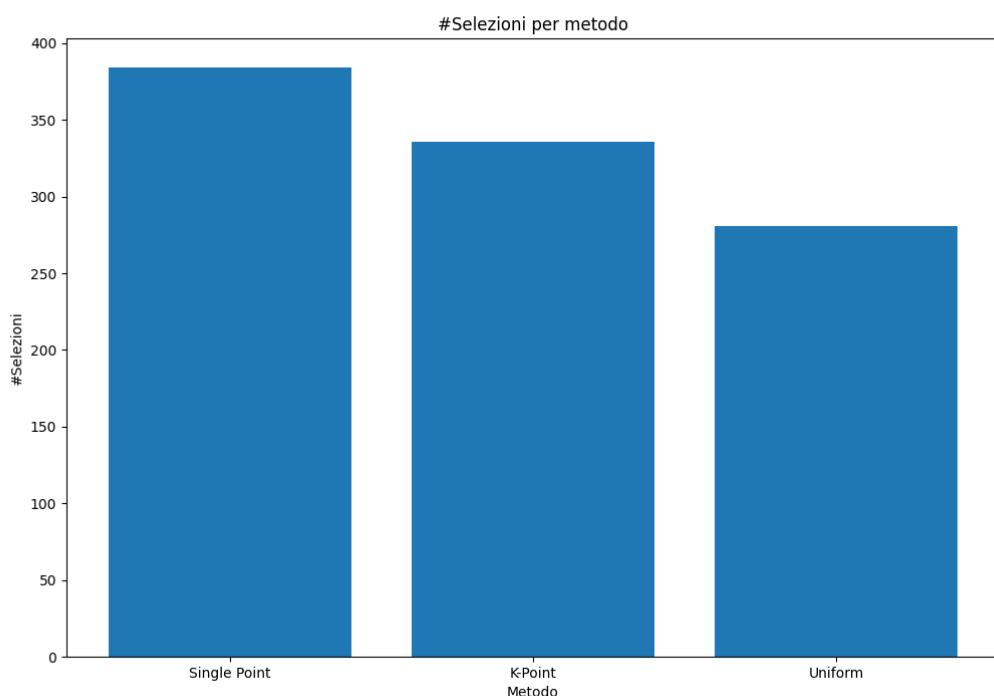
Istanza: 1000_-25_to_+25

Algoritmo: RandomGA

Run: 3/50



(a) Andamento delle probabilità



(b) # scelte per metodo

Figura 3.45: Scelta dei metodi di crossover:

Istanza: 1500_-25_to_+25

Algoritmo: RandomGA

Run: 50/50

3.3.5 Variazione delle probabilità di scelta dei metodi di crossover (interi esperimenti)

I grafici precedenti mostravano l'andamento della scelta dei diversi metodi di crossover nelle singole run. Come nel caso dei metodi di selezione, i dati delle singole run potrebbero essere fortemente influenzati dalla stocasticità dell'algoritmo, pertanto si riportano anche dei grafici che mostrano quante volte un singolo metodo di crossover è stato scelto nel totale delle 50 run per le singole istanze, relativi anche in questo caso solamente a 3 istanze, stavolta ovviamente relative solamente a RandomGA, che è l'unico tra i due algoritmi ad implementare la scelta dinamica del metodo di crossover.

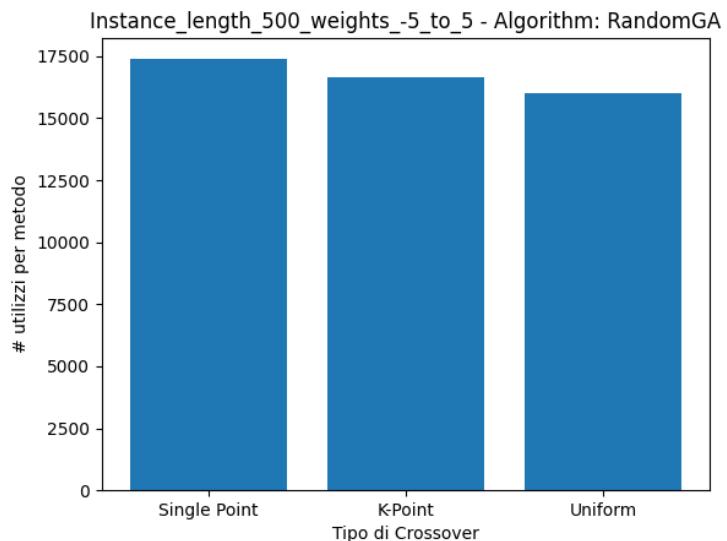


Figura 3.46: Scelta dei metodi di crossover (intera run):

Istanza: 500_-5_to_+5

Algoritmo: RandomGA

Valori esatti:

Single-Point	K-Point	Uniform
17335	16653	16012

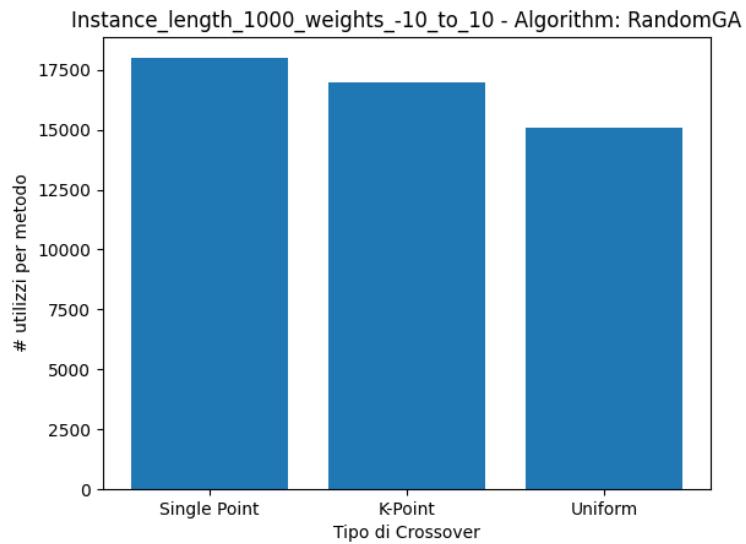


Figura 3.47: Scelta dei metodi di crossover (intera run):

Istanza: 1000_-10_to_+10

Algoritmo: RandomGA

Valori esatti:

Single-Point	K-Point	Uniform
17933	16980	15087

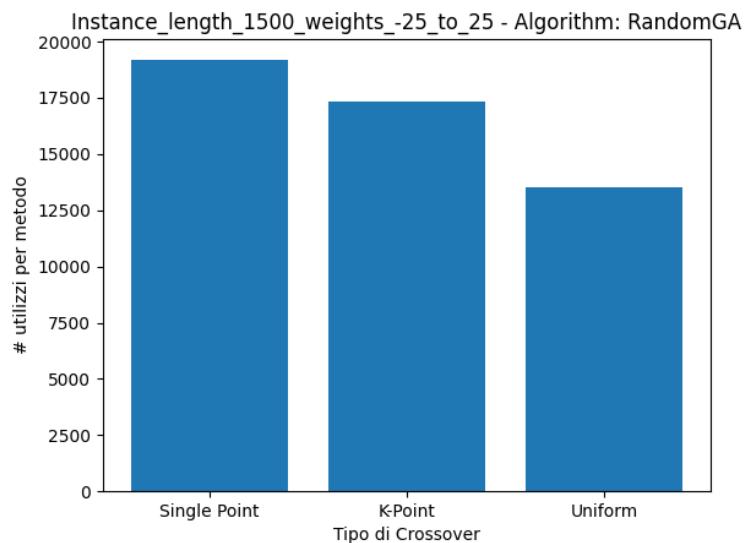


Figura 3.48: Scelta dei metodi di crossover (intera run):

Istanza: 1500_-25_to_+25

Algoritmo: RandomGA

Single-Point	K-Point	Uniform
19173	17286	13541

Come si può notare dai grafici precedenti, si evince che il metodo di crossover maggiormente usato tra quelli proposti è il **Single point crossover**.

D'altro canto, il metodo che viene invece scelto meno frequentemente è il metodo **Uniform Crossover**. Ciò è evidente osservando i grafici relativi all'andamento delle probabilità dei diversi metodi:

- le curva relativa al Single Point Crossover (curva blu) è quella che "si mantiene" su valori di probabilità elevati per la maggior parte delle iterazioni dell'algoritmo.
- al contrario, la curva relativa all'Uniform Crossover (curva verde) raggiunge spesso valori di probabilità bassi.

Si presume quindi dai risultati ottenuti che il metodo più conveniente per effettuare il crossover, al fine di aumentare la qualità delle soluzioni, sia il single point crossover.

Il perchè avviene ciò potrebbe essere dovuto al fatto che **il Single Point Crossover tende a mantenere intatti segmenti più lunghi delle soluzioni, il che permette di preservare soluzioni "buone" già presenti nella popolazione**.

Come detto anche nel caso dei diversi metodi di selezione, si noti che l'approccio probabilistico usato per la scelta dei metodi di crossover evita che l'algoritmo converga ad un unico metodo di crossover o, al contrario, vi sia un metodo di crossover mai usato dall'algoritmo.

Capitolo 4

Conclusione

Il presente progetto ha affrontato la variante del problema One Max proposta nella sezione introduttiva tramite l’implementazione di tre versioni di algoritmi genetici (GA classico, SelectionGA e RandomGA).

L’esecuzione degli algoritmi su 3 diverse tipologie di istanze, differenti in termini di lunghezza del vettore dei pesi da processare, ognuna delle quali a sua volta suddivisa in tre ”sottocategorie” di istanze, differenti per il range dei possibili valori dei pesi, ha fornito risultati interessanti e variegati in termini di qualità delle soluzioni trovate e di efficienza. Di seguito è proposta una sintesi delle principali osservazioni e delle conclusioni tratte dall’analisi dei tre algoritmi.

4.1 GA classico

L’algoritmo GA classico ha dimostrato di essere un algoritmo veloce ed efficiente su istanze con un vettore di pesi non molto lungo e con una variabilità di peso ridotta. Ad esempio, nell’istanza 500 con pesi variabili da -5 a +5, l’algoritmo ha funzionato abbastanza bene. Il principale vantaggio di questa versione dell’algoritmo è quello di avere tempi di esecuzione significativamente inferiori rispetto alle altre versioni, con un risparmio pari a circa il 20% rispetto a SelectionGA e del 60-65% rispetto a RandomGA.

Tuttavia, l’algoritmo GA classico presenta notevoli limiti quando viene applicato a istanze più lunghe e con un range di pesi più ampio. Negli esperimenti con un vettore dei pesi più lungo e con un range di pesi più ampio, come nell’istanza 1500 con pesi variabili da -25 a +25, l’algoritmo non ha restituito buoni risultati in termini di qualità della soluzione. In questo caso, l’errore rispetto alla soluzione ottimale è stato pari a circa il 24% sul valore della soluzione migliore trovata, con un errore medio pari a circa il 27%. Questo comportamento è dovuto principalmente alla scarsa diversità genetica nella popolazione durante le iterazioni avanzate, che porta a una riduzione della capacità esplorativa dell’algoritmo.

4.2 SelectionGA

Il SelectionGA rappresenta un miglioramento rispetto al GA classico, grazie all'introduzione di un meccanismo di selezione più sofisticato, basato sul concetto di Reinforcement Learning, che permette di utilizzare quattro diversi metodi di selezione degli individui, ovvero Tournament Selection, Roulette Selection, Elitism Selection e Rank Selection. Ciò aiuta a mantenere una maggiore diversità genetica nella popolazione. Questo algoritmo ha mostrato una buona performance su istanze con vettori dei pesi di dimensioni non troppo elevate, come quelle di lunghezza 500 e 1000, riuscendo a trovare la soluzione ottimale nella maggior parte dei casi. Quando il vettore dei pesi è più esteso, come nell'istanza 1500, SelectionGA non sempre riesce a trovare la soluzione ottimale, ma riesce comunque a trovare una soluzione che vi si avvicina molto, con un errore della migliore soluzione rispetto a quella ottimale pari a circa il 4%. Dal punto di vista dell'efficienza, i tempi di esecuzione di SelectionGA sono di poco superiori rispetto a GA classico, ma sono allo stesso tempo molto più bassi rispetto a RandomGA.

4.3 RandomGA

Il RandomGA, sebbene sia il più lento tra i tre algoritmi, ha dimostrato una notevole capacità di trovare soluzioni ottimali o molto vicine all'ottimo anche su istanze molto lunghe. Questo algoritmo si basa su un approccio che privilegia la diversità genetica a discapito della velocità, esplorando un ampio range di possibili soluzioni e riducendo il rischio di convergenza prematura.

Ciò è possibile in quanto l'algoritmo, oltre a variare il metodo di selezione come nell'algoritmo SelectionGA, implementa una variabilità anche nel metodo utilizzato per effettuare il crossover e dar vita a nuovi individui. In particolare, sono possibili tre tipologie di crossover: One-point Crossover, K-point Crossover, Uniform Crossover.

Nei test su istanze di lunghezza 500 e 1000, RandomGA ha trovato la soluzione ottimale, mentre su istanze di lunghezza 1500 ha ottenuto risultati subottimali che sono comunque estremamente vicini alla soluzione ottima, con un errore pari a circa il 0.1% tra la soluzione migliore trovata e la soluzione ottima. Tuttavia, la complessità computazionale del RandomGA è un aspetto critico da considerare

4.4 Considerazioni finali sulla qualità delle soluzioni

In conclusione, il progetto ha messo in luce i punti di forza e le debolezze di ciascuna variante di algoritmo genetico applicata alla variante del problema One Max proposta. Il GA classico, sebbene rapido, è limitato nella sua capacità di gestire istanze complesse e lunghe. Il SelectionGA offre un miglioramento significativo nella gestione della diversità genetica e nella capacità di trovare soluzioni ottimali su istanze di dimensioni piccole e medie, ma ha ancora margini di miglioramento per le istanze lunghe. Il RandomGA, con la sua eccellente capacità di trovare soluzioni ottimali anche su istanze molto lunghe, rappresenta un'ottima scelta quando la qualità della soluzione è critica, a patto di poter tollerare tempi di esecuzione elevati.

4.5 Considerazioni finali sui metodi di selezione e crossover

Oltre ad analizzare la qualità delle soluzioni trovate, sono anche stati analizzati i dati relativi ai metodi di selezione (in SelectionGA e RandomGA) e di crossover (in RandomGA), al fine di capire come questi ultimi influenzassero il raggiungimento di soluzioni più o meno ottimali. In particolare, la Roulette Selection e la Tournament Selection si sono dimostrati i metodi di selezione più scelti e quindi, presumibilmente, più efficaci nel migliorare la qualità delle soluzioni.

Nel caso poi di RandomGA, il Single Point Crossover ha prevalso come il metodo di crossover più vantaggioso, probabilmente perchè preserva maggiormente rispetto agli altri metodi di Crossover le soluzioni che risultano essere già buone.

L'approccio probabilistico adottato per la scelta dei metodi ha evitato una convergenza eccessiva su un singolo metodo, mantenendo una diversità necessaria per evitare convergenze ad ottimi locali e garantendo un'ampia esplorazione dello spazio delle soluzioni.

Dai risultati si può quindi concludere che **variare il metodo di selezione e, nel caso di RandomGA, anche il metodo di crossover, ha permesso di migliorare i risultati ottenuti dall'algoritmo genetico classico.**

Bibliografia

- [1] Claudia Cavallaro, Vincenzo Cutello, Mario Pavone, and Francesco Zito. Machine learning and genetic algorithms: A case study on image reconstruction. *Knowledge-Based Systems*, 284:111194, 2024.