Studio di algoritmi stocastici per la costruzione di quadrati magici

Metodi Computazionali della Fisica

Gabriele Bozzola
Matricola 882709

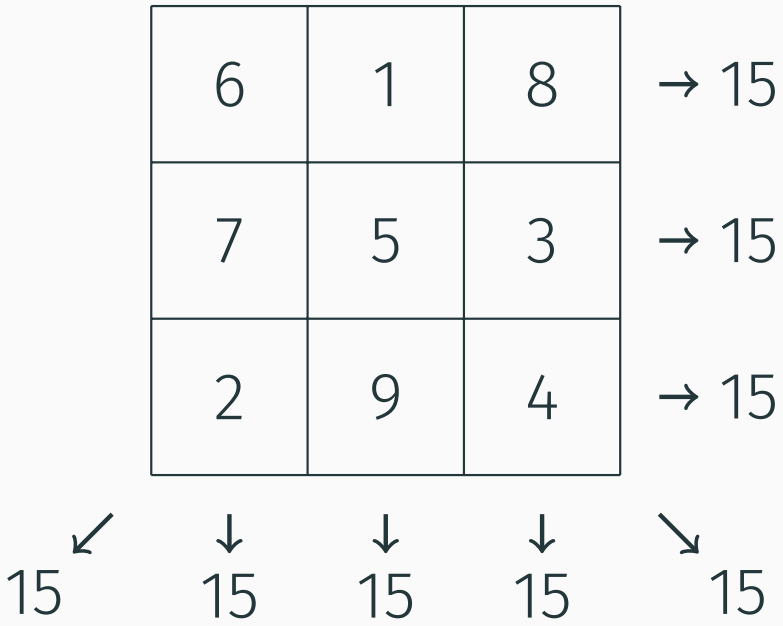
13 Luglio 2016

Università degli studi di Milano

QUADRATI MAGICI

Definizione

Un quadrato magico normale $N \times N$ è una matrice di ordine N contenente tutti i numeri naturali da 1 a N^2 tali che la somma di tutti gli elementi sulle righe, sulle colonne e sulle diagonali sia sempre la stessa, detta numero magico.



QUADRATI MAGICI – PROPRIETÀ

Teorema di esistenza

$\forall N \in \mathbb{N} - \{2\}$ è sempre possibile costruire almeno un quadrato magico normale.

Formula per la costante magica

$\forall N \in \mathbb{N} - \{2\}$ la costante magica *m.v.* di un quadrato di ordine N è:

$$m.v. = \frac{1}{2} N(N^2 + 1)$$

QUADRATI MAGICI – NUMERO

Numero di quadrati magici di ordine N

Il numero di quadrati magici è noto con precisione solo per $N < 6$.
La percentuale sul totale dei possibili quadrati tende a zero per N che tende a $+\infty$.

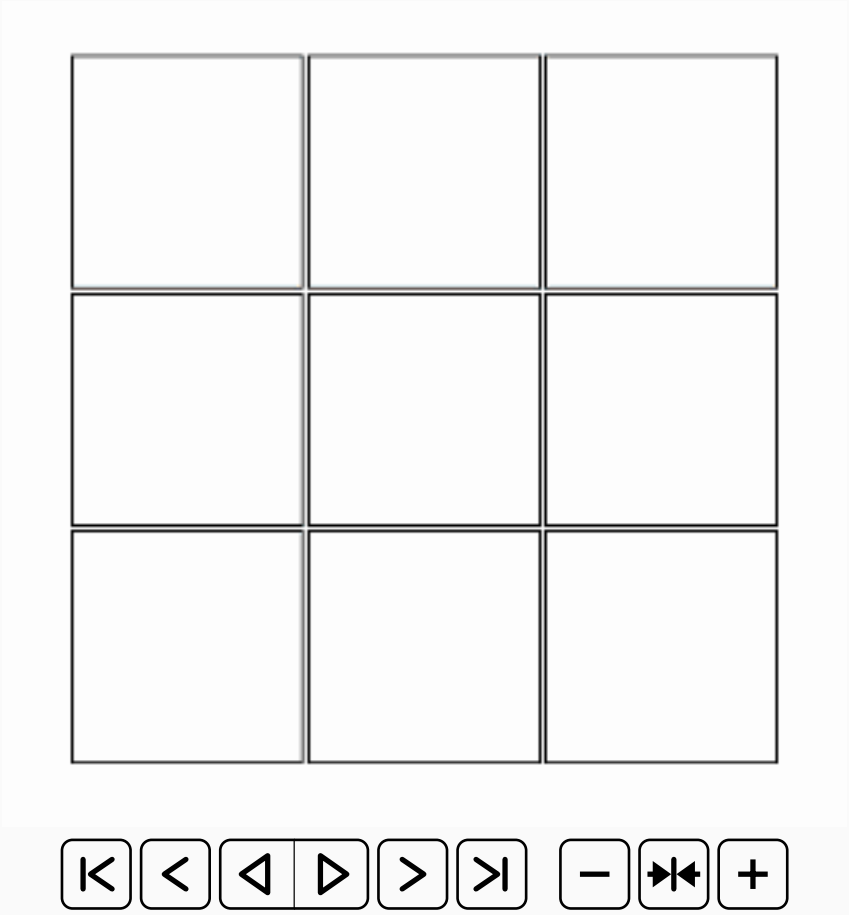
N	N_{ms}	N_{ns}	%
2	0	$\sim 10^{1}$	0
3	1	$\sim 10^{5}$	$\sim 10^{-5}$
4	880	$\sim 10^{12}$	$\sim 10^{-7}$
5	275 305 224	$\sim 10^{24}$	$\sim 10^{-18}$
6	$\sim 10^{19}$	$\sim 10^{41}$	$\sim 10^{-22}$
20	$\sim 10^{744}$	$\sim 10^{868}$	$\sim 10^{-124}$
35	$\sim 10^{2992}$	$\sim 10^{3252}$	$\sim 10^{-250}$
50	$\sim 10^{7000}$	$\sim 10^{7410}$	$\sim 10^{-410}$

Trovare quadrati magici è difficile.*

*Dati ottenuti con metodi statistici da Trump W. e con errore inferiore a 1 %.

QUADRATI MAGICI - METODI DI COSTRUZIONE DETERMINISTICI

- Per quadrati di ordine dispari: metodo de la Loumbre, metodo di Conway, metodo Pheru, ...
- Per quadrati pari: metodo Medjing, ...
- Per quadrati singolarmente pari (*N* è multiplo di quattro): metodo LUX, metodo Strachey, ...



Quadrati magici - problemi di questi metodi

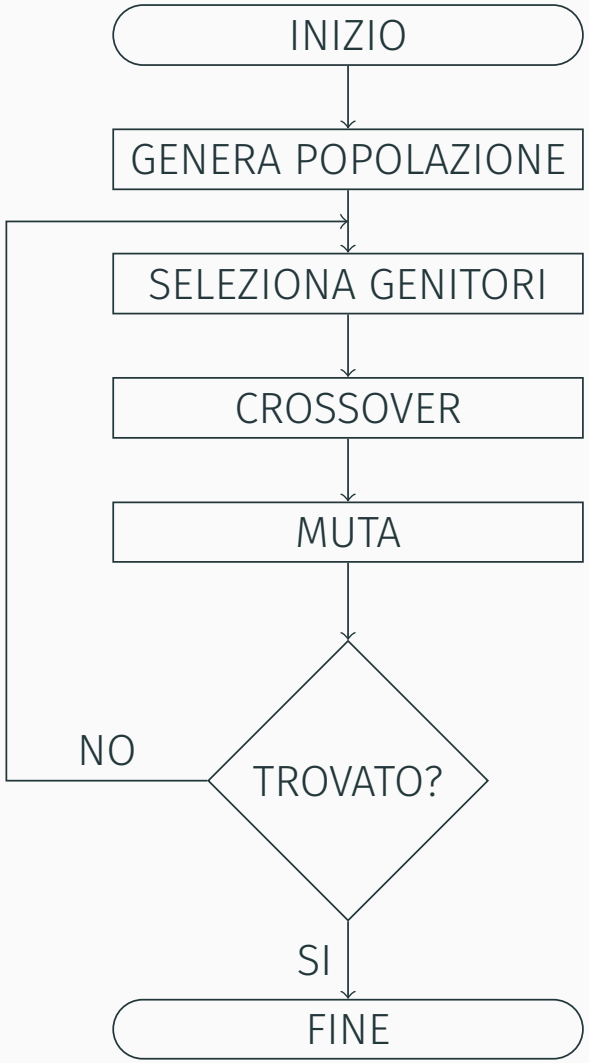
- Costruiscono sempre il medesimo quadrato
- Non sono generalizzabili ad altri tipi di quadratici magici

I metodi stocastici risolvono questi problemi.

ALGORITMO GENETICO - FUNZIONAMENTO

Gli algoritmi genetici implementano il principio darwiniano di sopravvivenza del più adatto.
Gli individui:

- sono possibili soluzioni del problema
- sono classificati in base alla loro fitness, che quantifica quanto si avvicinano alla soluzione reale
- si riproducono in maniera sessuata (crossover)
- possono subire mutazioni



Algoritmo genetico – perché è adatto?

Perché il problema della costruzione dei quadrati magici è un buon problema da affrontare con gli algoritmi genetici?

- Lo spazio delle soluzioni è estremamente vasto
- I quadrati possono essere codificati in modo diretto come individui
- Il problema può essere formulato come ottimizzazione di una funzione di fitness

ALGORITMO GENETICO – IMPLEMENTAZIONE

Per implementare un algoritmo genetico bisogna a pensare a:

- Che funzione di fitness utilizzare?
- Come far selezionare i genitori?
- Come far riprodurre i quadrati?
- Come mutarli?

ALGORITMO GENETICO – FITNESS

Funzioni di fitness implementate

- `totalSquared`: somma dei quadrati delle discrepanze delle somme di ogni linea dal valore magico
- `totalAbs`: somma dei moduli delle discrepanze delle somme di ogni linea dal valore magico
- `correctLines`: numero di linee magiche

ALGORITMO GENETICO – METODI DI SELEZIONE

Metodi di selezione implementati

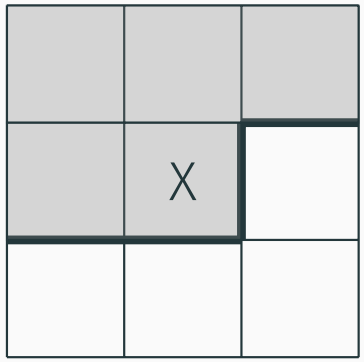
- `fitnessProportionate`: probabilità di selezione proporzionale alla fitness
- `similarSquare`: probabilità di selezione dipendente dalla fitness e dalla distanza dal quadrato migliore
- `fittests`: alcuni individui non si riproducono, gli altri hanno uguale probabilità di selezione
- `elitism`: alcuni individui passano direttamente alla generazione successiva, i restanti vengono selezionati secondo uno dei precedenti metodi

Metodi di crossover

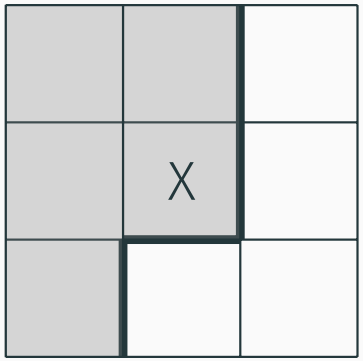
Crossover a uno o due punti verticale o orizzontale.
Se il crossover produce numeri doppi questi vengono sistemati casualmente.

Metodi di mutazione

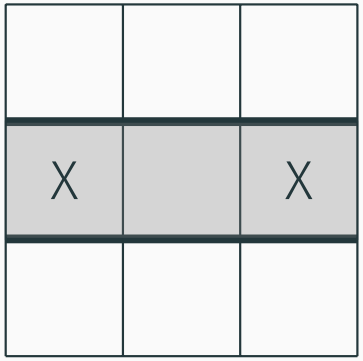
- Scambio di una coppia
- Scambio di due colonne
- Scambio di due righe
- Permutazione di una riga
- Permutazione di una colonna



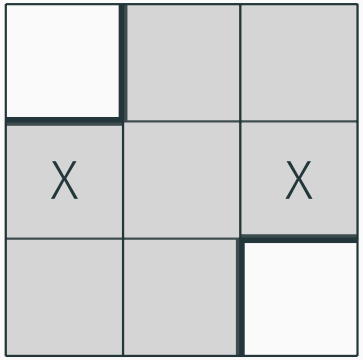
(a) Crossover orizzontale



(b) Crossover verticale



(c) Crossover a due punti orizzontale



(d) Crossover a due punti verticale

ALGORITMO GENETICO – RISULTATI

Nessuna combinazione di fitness e metodi di selezione e crossover è riuscita a costruire quadrati di dimensioni superiori a 3×3!

Motivo: lo spazio delle soluzioni non è connesso rispetto a nessuna funzione di fitness.

ALGORITMO GENETICO – POSSIBILI MIGLIORAMENTI

- I crossover sono dannosi per giungere ad una soluzione. Non si possono eliminare?
- Non si può *aiutare* l'algoritmo a convergere sbloccandolo nei momenti di stallo?

ALGORITMO EVOLUTIVO - FUNZIONAMENTO

Gli algoritmi evolutivi sono particolari algoritmi genetici in cui:

- Non ci sono crossover
- Le mutazioni sono molto più sofisticate
- Sostanzialmente si lavora con un solo individuo

Algoritmo di Xie e Kang

L'algoritmo di Xie e Kang* è un algoritmo evolutivo per la costruzione di quadrati magici normali con:

- Mutazioni dinamiche e adattive
- Rettificazioni locali
- Congettura della costruzione a due fasi

Congettura della costruzione a due fasi

Un quadrato semimagico è sempre completabile ad un quadrato magico utilizzando un numero finito di permutazioni di righe e di colonne oppure di rettificazioni locali.

*Xie, T. e Kang, L. (2003), *An Evolutionary Algorithm for Magic Squares*, The 2003 Congress on Evolutionary Computation, 2003.

Algoritmo di Xie e Kang – codifica dell'individuo

Individuo

Un individuo è una coppia di matrici (M, Σ), la prima è il quadrato da rendere magico, la seconda contiene informazioni necessarie per le mutazioni.

Fitness

$$f(M) = \begin{cases} \sum_{i=1}^{N} (\text{row}(i) + \text{col}(i)) & \text{semimagico} \\ -(\text{dg1} + \text{dg2}) & \text{altrimenti} \end{cases}$$

Dove col(i) e row(j) sono rispettivamente la somma degli elementi sulla $i-$esima colonna e $j-$esima riga di M, e dg1 e dg2 sono la somma degli elementi sulla diagonale e sull'antidiagonale di M.

Mutazioni

- Dinamiche: la probabilità di mutazione non è fissa, ma dipende dal numero di linee non magiche
- Adattive: le mutazioni dipendono da quanto il quadrato non è magico

Mutazioni

- Mutazioni puntuali per quadrati generici
- Mutazioni lineari per quadrati che hanno solo le diagonali non magiche (quadrati semimagici)

Algoritmo di Xie e Kang – mutazioni puntuali I

Insiemi di mutazione

- S_1 numeri la cui riga e colonna non è magica
- S_2 numeri in righe o colonne non magiche

Mutazioni

Siano n_{col} e n_{row} il numero di colonne e di righe non magiche. Le mutazione sono scambi di numeri tra:

- S_1 e S_2 con probabilità $1/(n_{row}n_{col})$.
- S_2 e S_2 con probabilità P_M.
- S_2 e M con probabilità P_M.

dove

$$P_M(x) = \begin{cases} 1/n_{row} & \text{se} \quad x \text{ è in una riga non magica} \\ 1/n_{col} & \text{se} \quad x \text{ è in una colonna non magica} \\ 1/\left(n_{row}n_{col}\right) & \text{se} \quad x \text{ è in entrambe} \end{cases}$$

Algoritmo di Xie e Kang – mutazioni puntuali II

Esempio: S_1 in S_2

Siano $m_{ij} \in M$ e $\sigma_{ij} \in \Sigma$

1. Calcolo $new = m_{ij} + \text{randint}(-\sigma_{ij}, \sigma_{ij})$
2. Aggiusto se è invalido:

$$\begin{cases} new = \text{randint}(1, N) & \text{se} \quad new < 1 \\ new = N^2 - \text{randint}(0, N) & \text{se} \quad new > N^2 \end{cases}$$

3. Cerco l'elemento in S_2 che più si avvicina a new cioè $t \in S_2$ tale che soddisfi $\min_{t \in S_2} |new - t|$.
4. Scambio t e new in M.

Algoritmo di Xie e Kang – mutazioni puntuali III

Esempio: S_1 in S_2

5. Calcolo $z = \sigma_{ij} + \text{randint}(-1, 1)$
6. Aggiusto se è invalido:

$$z = \text{randint}(1, \sigma_t) \quad \text{se} \quad z < 1 \quad \text{o} \quad z > \sigma_t$$

con:

$$\sigma_t = \begin{cases} |f(M)| / (n_{row} + n_{col}) & \text{se} \quad n_{row} n_{col} \neq 0 \\ |f(M)| / n_{diag} & \text{se} \quad n_{row} n_{col} = 0 \end{cases}$$

σ_t piccola: quadrato quasi magico

7. Sostituisco a σ_{ij} in Σ il valore z.

Algoritmo di Xie e Kang – mutazioni lineari

Mutazioni lineari

Sono permutazioni casuali di una linea di un quadrato semimagico.

1. Seleziono un numero q intero da 1 a N.
2. Per q volte estraggo una linea.
3. La sostituisco con una permutazione casuale dei suoi elementi.

La linea rimane magica.

Rettificazioni locali

L'algoritmo potrebbe rimanere in una fase di stallo, per questo conviene operare con approcci sistematici:

- Rettificazioni lineari: cercano di aumentare il numero di linee magiche.
- Rettificazioni diagonali: cercano di rendere le diagonali di un quadrato semimagico magiche.

Le rettificazioni sono ottenute analizzando tutto il quadrato in cerca di tutte le coppie o i quartetti tali che una loro permutazione migliori il quadrato.

Algoritmo di Xie e Kang – rettificazioni locali lineari

Esempio di rettificazione locale lineare

Due numeri m_{ks} e m_{ls} sono scambiati alla riga k e l e alla colonna s se sono soddisfatte:

- $\text{row}(k) - m.v. = m_{ks} - m_{ls}$
- $m.v. - \text{row}(l) = m_{ks} - m_{ls}$

con $m.v.$ costante magica.

Sono state implementate altre tre condizioni simili, che coinvolgono due o quattro numeri.

1	5	6	→ 12
4	3	8	→ 15
2	7	9	→ 18

(e) Prima

1	5	**9**	→ 15
4	3	8	→ 15
2	7	**6**	→ 15

(f) Dopo

Algoritmo di Xie e Kang – rettificazioni locali diagonale

Rettificazioni locali diagonali:

- Puntuali: se scambiano numeri
- Lineari: se scambiano linee

Esempio di rettificazione locale diagonale puntuale

Se sono soddisfatte le condizioni:

- $a_{ii} + a_{ij} = a_{ji} + a_{jj}$
- $(a_{ii} + a_{jj}) - (a_{ij} + a_{ji}) = \mathrm{dg1} - m.v.$

allora a_{ii} è scambiato con a_{ji} e a_{ji} con a_{jj}.

4	1	9
7	2	6
8	5	3

↘ 9

(g) Prima

4	1	9
7	**5**	**3**
8	**2**	**6**

↘ 15

(h) Dopo

Metodi di selezione

Il nuovo genitore è

- $(\mu, \lambda) - ES$: il migliore figlio della precedente. Permette maggiore variabilità.
- $(\mu + \lambda) - ES$: il migliore tra il genitore e i figli della precedente. Permette di conservare i risultati ottenuti.

I metodi applicati dipendono dalle caratteristiche del quadrato migliore.

algoritmo di Xie e Kang – alcuni punti delicati o interessanti I

Funzione di fitness:

```
(*La fitness e' negativa quando il quadrato e'
semimagico, questo mi permette di renderli preferiti
ai quadrati generici*)
If[incorrectLines[ind] === 0,
   Return[-Total[diagonalsDeviation[ind]]],
   Return[Total[linesDeviation[ind]]];
];
```

Estensione dei metodi su tutta la popolazione:

```
fitnessPop[pop_List] := Return[fitness /@ pop];
```

algoritmo di Xie e Kang – alcuni punti delicati o interessanti II

`fittestChild`: selezionare l'individuo migliore in **pop**.

```
fp = fitnessPop[pop];
(*Controllo se c'e' l'individuo perfetto*)
If[Length[Position[fp, 0]] =!= 0,
   Return[pop[[Position[fp, 0][[1,1]]]]];
];
(*Io voglio l'individuo piu' vicino a zero,
ma voglio anche privilegiare chi ha fitness negativa*)
If[Min[fp] < 0,
   fp = (#)^(-1) & /@ fp
];
(*Posizione del migliore*)
Random[Integer, {1, Length[Position[fp, Min[fp]]]}]
```

algoritmo di Xie e Kang – alcuni punti delicati o interessanti III

Algoritmo completo:

```
[...] (*Vari controlli*)
offspring = ParallelMap[mutate, offspring];
[...] (*Altri controlli*)
[...] (*Salvo risultati intermedi*)
```

L'algoritmo completo esegue operazioni che coinvolgono la popolazione intera parallelizzate su una macchina (lara) dotata di quattro core.

Algoritmo di Xie e Kang – risultati

Risultati con N ordine del quadrato, n_{tent} numero di tentativi di esecuzione, n_{ok} numero di successi e τ tempo medio di esecuzione.

N	n_{tent}	n_{ok}	τ
3	10	10	0.12 s
10	10	10	55 s
15	10	10	5.75 min
20	10	10	31.2 min
25	10	10	1.73 h
30	10	10	4.23 h
35	10	10	12.38 h
40	10	10	25.79 h

(a) Popolazione di 25 figli.

N	n_{tent}	n_{ok}	τ
3	10	10	0.22 s
10	10	10	69 s
15	10	10	4.37 min
20	10	10	26.3 min
25	10	10	1.51 h
30	10	10	3.97 h
35	10	10	8.63 h
40	10	10	15.73 h

(b) Popolazione di 10 figli.

Algoritmo di Xie e Kang – risultati

Risultati con N ordine del quadrato, n_{tent} numero di tentativi di esecuzione, n_{ok} numero di successi e τ tempo medio di esecuzione.

N	n_{tent}	n_{ok}	τ
3	10	10	0.12 s
10	10	10	55 s
15	10	10	5.75 min
20	10	10	31.2 min
25	10	10	1.73 h
30	10	10	4.23 h
35	10	10	12.38 h
40	10	10	25.79 h

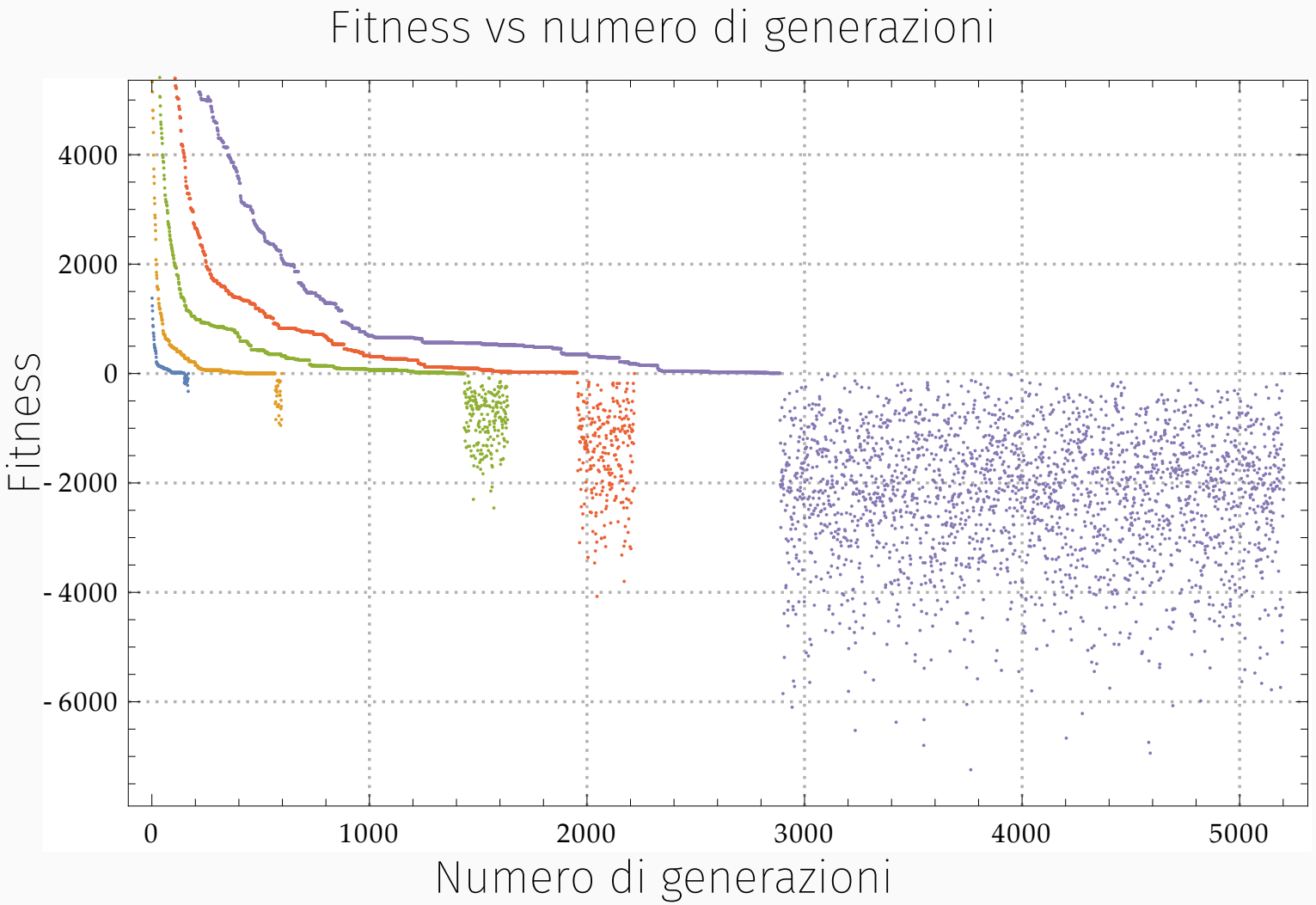
(c) Popolazione di 25 figli.

N	n_{tent}	n_{ok}	τ
3	10	10	0.22 s
10	10	10	69 s
15	10	10	4.37 min
20	10	10	26.3 min
25	10	10	1.51 h
30	10	10	3.97 h
35	10	10	8.63 h
40	10	10	15.73 h

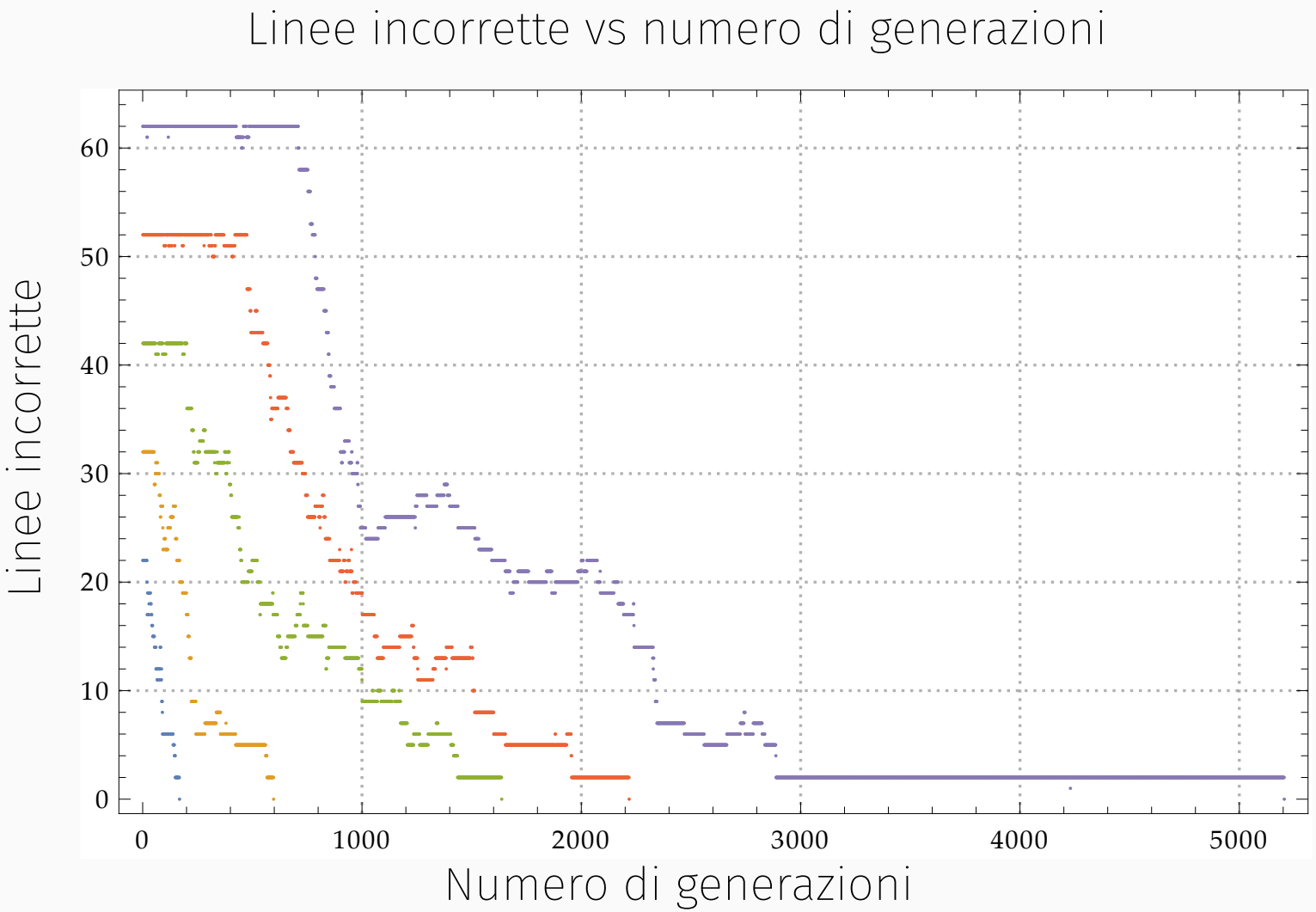
(d) Popolazione di 10 figli.

L'algoritmo non ha mai fallito.

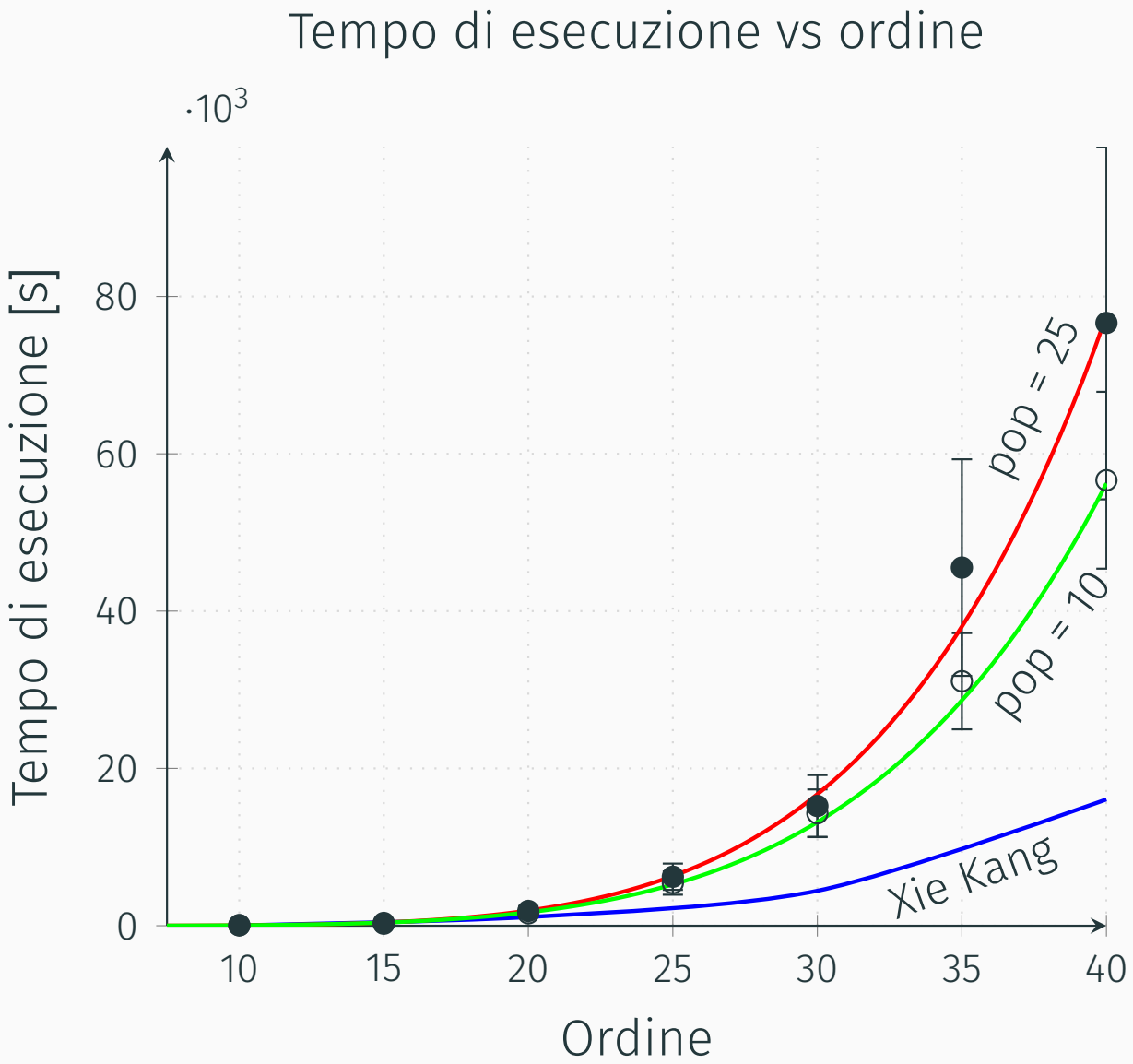
Algoritmo di Xie e Kang – fitness



Algoritmo di Xie e Kang – linee non magiche



Algoritmo di Xie e Kang – tempi di esecuzione



algoritmo di Xie e Kang – distribuzione dei tempi di esecuzione I

Distribuzione dei tempi di esecuzione

Routine	Tempo speso (%)
`selectFittest`	~ 0.01
`mutate`	~ 0.7
`rectifyDiagonals`	~ 7.7
`rectifyLines`	~ 91
/ \	/ \
`onePair` | `twoPairs`	~ 13 | ~ 87

Algoritmo di Xie e Kang – distribuzione dei tempi di esecuzione II

Perché? (Rettificazioni a due coppie)

```
(*i1,j1 indici del primo elemento*)
For[i1 = 1, i1 <= N, i1++,
    For[j1 = 1, j1 <= N, j1++,
        (*Scorro il resto del quadrato dopo i1,j1*)
        For[i2 = i1 + 1, i2 <= N, i2++,
            For[j2 = 1, j2 <= N, j2++,
                [...]  (*Vari controlli*)
```

Il numero di operazioni cresce come N^4.
E' necessaria questa implementazione perché bisogna operare direttamente con gli indici.

- E’ possibile realizare con successo algoritmi stocastici per la costruzione di quadrati magici se non si usano crossover e se si interviene in modo sistematico
- Questi algoritmi sono molto più efficienti della ricerca a tappeto
- Si è mostrato che l’approccio di Xie e Kang funziona
- Non si è ritrovata la legge di scala trovata da Xie e Kang, probabilmente a causa della quasi totale ignoranza riguardo l’implementazione originale

Conclusioni e sviluppi futuri – Generalizzazioni e sviluppi futuri

Alcune questioni lasciate aperte:

- Ottimizzare implementazione, `Compile[]`?
- Confrontare con realizzazione in linguaggio compilato
- Parallelizzare i metodi di rettificazione
- Indagare dipendenza del tempo di esecuzione dalla dimensione della popolazione

Possibili estensioni: E' possibile generalizzare l'algoritmo fintato che si generalizzano i metodi di rettificazione, quindi per tutti quei casi in cui $m.v.$ è fissato. Ad esempio:

- Quadrati magici vincolati
- Quadrati magici non normali con costante magica fissata

Mathematica non si è rivelato necessario nell'implementazione di questi algoritmi perché la quasi totalità della manipolazione è numerica e non simbolica.
Tuttavia Mathematica ha reso l'implementazione più diretta perché:

- Le funzioni vettorializzate, come `Map[]`, `Apply[]`, `Table[]` permettono l'implementazione molto elegante dei metodi che agiscono sulle popolazioni intere
- `Replace[]` permette di fare gli scambi in modo conciso è chiaro

Il costo di questa semplificazione è probabilmente la riduzione nell'efficienza.

GRAZIE PER L’ATTENZIONE!