Studio di algoritmi stocastici per la costruzione di quadrati magici

Gabriele Bozzola

Università degli Studi di Milano
bozzola.gabriele@gmail.com

Luglio 2016

Sommario

In questo lavoro vengono presentati due algoritmi stocastici per costruire quadrati magici normali, che sono matrici di numeri naturali distinti che godono della proprietà che la la somma degli elementi su ciascuna riga e colonna e sulle diagonali è sempre la stessa, detta costante magica. La prima implementazione è un algoritmo genetico realizzato utilizzando diverse funzioni di fitness e metodi di selezione, mentre la seconda consiste in un algoritmo evolutivo, basato sul lavoro di Xie e Kang [10]. Nessuna delle implementazioni dell'algoritmo genetico realizzate in questo lavoro ha portato alla costruzione di quadrati magici, a causa dell'impossibilità di formulare in modo sufficientemente adeguato il problema. L'algoritmo evolutivo, al contrario, si è mostrato efficacie nella costruzione di quadrati magici. Si sono quindi confrontati i risultati ottenuti con quelli originali di Xie e Kang, trovando un disaccordo in merito a come scalano i tempi di esecuzione all'aumentare dell'ordine del quadrato. Tale discostamento è probabilmente dovuto ai limiti dell'implementazione in Mathematica e all'ignoranza su come i risultati originali sono stati effettivamente ottenuti. In ogni caso questo lavoro dimostra come sia possibile implementare un algoritmo stocastico per un complicato problema di ordinamento in cui non è possibile costruire una fitness adeguata. Infine, alcune possibili modalità per migliorare l'attuale implementazione sono discusse.

I. Introduzione

Un quadrato magico di ordine N è una matrice quadrata M di dimensioni $N \times N$ contenente numeri naturali distinti disposti in modo tale che la somma dei valori su ciascuna riga, colonna e diagonale sia sempre la stessa, detta *costante o numero magico*. Qualora i numeri che compaiono sono i primi N^2 allora il quadrato è detto *normale*, e la somma che devono le linee devono soddisfare è nota a priori in quanto esiste una relazione algebrica che lega l'ordine di un quadrato magico normale con la costante magica: per una matrice di ordine N è:

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

La dimostrazione di questa elementare proprietà è riportata in appendice. Si dimostra inoltre che, ad eccezione di $N = 2$, è sempre possibile costruire almeno un quadrato magico per ogni ordine. Un esempio di quadrato magico normale è riportato in figura 1.

6	1	8	→ 15
7	5	3	→ 15
2	9	4	→ 15
↙ 15	↓ 15	↓ 15	↓ 15 ↘ 15

Figura 1: *Esempio di quadrato magico 3×3 in cui sono riportati i valori delle somme di tutte le righe, le colonne e le diagonali. A meno di riflessioni non ne esistono di differenti.*

I quadrati magici hanno una storia molto lunga, iniziata probabilmente in Cina nel settimo secolo prima di Cristo, quando ai quadrati magici erano attribuite proprietà mistiche e rituali. Dalla Cina passarono in India e successivamente in Europa, dove furono continuamente considerati come oggetti dotati di poteri magici, il che spiega il loro utilizzo in discipline come l'alchimia o l'astrologia. Il primo studio matematicamente ponderato dei quadrati magici fu condotto da Simon De la Loubère alla fine del Seicento, che fornì anche un metodo di costruzione per alcuni tipi di quadrati. Da quel momento numerosi matematici si dedicarono allo studio delle proprietà dei quadrati magici, tra cui Euler e Lagrance e Lucas, anche se rimangono aperti molti interrogativi [1]. Ad oggi sono in palio 8000 $ e diverse bottiglie di champagne per chi risolve alcuni di questi enigmi [3].

I quadrati magici hanno inoltre alcune applicazioni tecnologiche, tra cui nella crittografia [8], nella steganografia [7] (la tecnica di occultare informazioni nelle immagini), ma anche in teoria dei grafi o dei giochi, e in molti altri campi tra cui la fisica [5].

Costruire quadrati magici non è un compito semplice, in quanto il loro numero è molto piccolo rispetto a tutte le possibilità e le equazioni che definiscono la *magicità* di un quadrato non sono abbastanza stringenti per utilizzare approcci di forza bruta. Anche se a oggi non esiste ancora una formula che permetta di calcolare il numero di quadrati magici di ordine N, si possono effettuare stime sono utilizzando metodi Monte Carlo e approcci con tecniche di meccanica statistica. I risultati di queste ricerche sul numero di quadrati magici sono riportate in tabella 1, dal quale risulta evidente che la magicità è una proprietà molto rara.

Tabella 1: *Stime del numero di quadrati magici distinti N_{ms} al variare dell'ordine N dei quadrati magici normali. N_{ns} è il numero di possibili quadrati magici normali distinti costruibili all'ordine N. In questo caso due quadrati sono ritenuti distinti se non è possibile trasformare l'uno nell'altro utilizzando riflessioni. I presenti valori sono dovuti a Trump [9] e sono ottenuti con metodi di Monte Carlo Backtracking con errori intorno a* 1 %.

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	$\sim 10^{7000}$	$\sim 10^{7410}$	$\sim 10^{-410}$

i. Approcci deterministici

Nonostante il numero di quadrati magici sia piccolo sono disponibili alcuni metodi di costruzione da molti anni, i più semplici dei quali sono già stati elencati da Kraitchik [4] nel 1942.

Per quadrati dispari è disponibile il metodo Siamese[1] (anche noto come metodo di de la Loumbre), il metodo "a losanghe" di Conway e il metodo Pheru. Per i quadrati di ordine pari esiste il metodo Medjig, mentre per quelli singolarmente pari, cioè di ordine $4n + 2$ con $n \in \mathbb{N}$ sono noti il metodo LUX di John Conway e il metodo Strachey.

Questi metodi sono di due tipologie: costruzioni passo a passo, oppure completamenti. Nel primo caso si inizia posizionando uno o più numeri, e con passaggi ricorsivi (detti movimenti) si utilizzano i valori già posizionati nel quadrato per posizionarne uno successivo. Nel secondo caso si paro da una soluzione di ordine N per giungere ad una di $N + 1$ utilizzando apposite trasformazione, come nel metodo Strachey.

ii. Approcci stocastici

Gli algoritmi elencati nella sezione precedente sono stati implementati in programmi di computer e risultano sufficientemente efficienti per la costruzione di quadrati magici. Tuttavia,

[1] Simon de la Loubère è stato ambasciatore francese in Siam, luogo dove ha appreso il metodo che porta il suo nome.

tutti questi programmi hanno una limitazione: generano sempre il medesimo quadrato fissato l'ordine. Per questo motivo tali algoritmi risultano poco generalizzabili e quindi inadatti per studiare quadrati che, oltre ad essere magici, godono di ulteriori proprietà (ad esempio i quadrati bimagici, oppure i quadrati magici vincolati). A causa di ciò è conveniente provare ad implementare algoritmi in cui ci siano fattori di casualità che permettono di costruire sempre nuovi quadrati magici. In letteratura si trovano già alcuni risultati sulla costruzione di quadrati magici con questo tipo di metodologie, e i più efficienti algoritmi stocastici attualmente realizzati sono basati su tecniche note come *metodi iper-euristici* che permettono di realizzare un quadrato magico vincolato di dimensioni 2600×2600 in meno di un minuto [2]. Questi algoritmi sono generalmente molto complessi e articolati, mentre in questo lavoro sono state implementate due soluzioni meno sofisticate: alcuni algoritmi genetici e uno evolutivo. Questi sono stati realizzati in Mathematica 8, con la scopo principale di ottenere dei programmi che effettivamente sono in grado di costruire dei quadrati magici. Gli algoritmi che riescono a convergere sono quindi stati caratterizzati in termini delle leggi di scala in funzione della dimensione dell'input e sono stati confrontati con i risultati presenti in letteratura. Si è utilizzata la versione 8 di Mathematica principalmente perché permette di parallelizzare alcune funzioni in modo estremamente semplice, come ad esempio con il comando `ParallelTable[]` o il comando `ParallelMap[]`, che sono i corrispettivi parallelizzati di `Table[]` e `Map[]`, girando su quattro core questo ha permesso di ridurre di circa il 75 % i tempi di esecuzione rispetto ad un implementazione puramente sequenziale. Questi ed altri dettagli saranno discussi successivamente.

iii. Terminologia

In questo lavoro si utilizzerà il termine *linea* indicando in modo generico una riga o una colonna, laddove i termini riga e colonna designano i medesimi concetti generalmente utilizzati in algebra matriciale. Una linea si dice *magica* se la somma dei numeri che la compongono è il numero magico.

Si compie inoltre un innocuo abuso di linguaggio utilizzando la parola quadrato intendendo quella che matematicamente è una matrice quadrata.

Altra terminologia e notazione verrà introdotta pian piano.

II. Algoritmi genetici

Gli algoritmi genetici sono metodi di ottimizzazione stocastici che implementano le modalità dell'evoluzione biologica. Questi algoritmi utilizzano una popolazione di possibili soluzioni del problema (chiamati *individui*) cercando di trovare quella che meglio approssima la vera soluzione del problema con il principio darwiniano di sopravvivenza del migliore. Tutti gli algoritmi genetici si basano su un ciclo di selezione - riproduzione - mutazione.

Selezione Inizialmente si selezionano i migliori individui della popolazione secondo un parametro che è chiamato *fitness*, la quale è una funzione dei *geni*, ovvero delle caratteristiche della soluzione, che è generalmente codificata come una stringa di numeri. Gli individui con fitness più alta hanno maggiore probabilità di essere selezionati per la riproduzione.

Riproduzione Dopo la selezione si producono figli a partire da coppie degli individui selezionati precedentemente. I figli sono generati simulando quella che è la riproduzione sessuata in biologia cioè con lo scambio di geni dei genitori (questo processo è noto come *crossover*). Questa fase permette di costruire individui che hanno le caratteristiche migliori di entrambi i genitori in modo da avvicinarsi alla soluzione.

Mutazione Infine vengono applicate delle mutazioni sui figli che ne alterno casualmente qualche caratteristica. In questo modo ci si assicura di esplorare tutto lo spazio delle soluzioni è di non fossilizzarsi in un massimo locale della funzione di fitness.

Una volta che si è completato il ciclo si verifica se si è trovata la soluzione del problema, altrimenti i figli della presente generazione diventa i genitori della successiva e il ciclo si ripete. Il workflow base di un algoritmo genetico è riportato in figura 2.

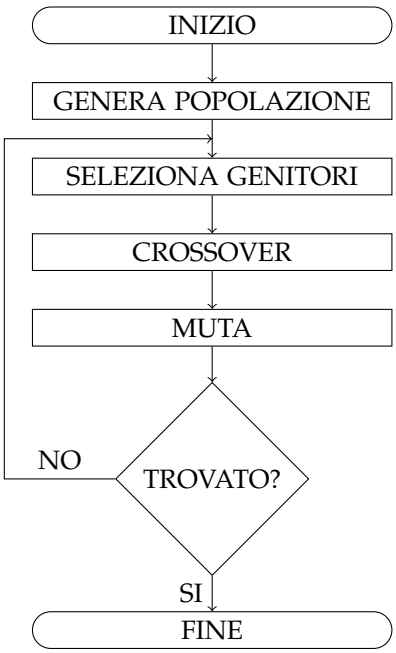


Figura 2: *Diagramma di flusso di un algoritmo genetico. L'algoritmo può non convergere e rimanere bloccato nel loop.*

Il problema della costruzione di quadrati magici ha alcune delle caratteristiche adatte per essere affrontato con un algoritmo genetico:

- Lo spazio delle soluzioni è estremamente vasto, consistendo nelle permutazioni di N^2 elementi, sottoposte a $2N-1$ vincoli (uno per ogni riga, per ogni colonna e per le due diagonali). Lo spazio delle soluzioni ha quindi $(N^2-2N-1)!$ elementi (a titolo di esempio per $N=9$ il numero di soluzioni possibili è più grande del numero di atomi presenti nell'universo osservabile).
- La complessità del problema è fortemente NP, quindi tecniche di *brute force* non sono attuabili.
- I quadrati magici possono essere codificati in modo diretto in individui dell'algoritmo genetico.
- Il problema può essere formulato come un problema di ottimizzazione di una funzione di fitness.

Sono stati implementati diversi algoritmi genetici, i quali differiscono tra loro principalmente per la scelta della funzione di fitness e del metodo di selezione.

i. Funzioni di fitness

Le funzioni di fitness che sono state implementate in questo lavoro sono:

- `totalSquared`: La fitness di un individuo è data dalla somma dei quadrati delle differenze di ogni linea dal numero magico.
- `totalAbs`: La fitness di un individuo è data dalla somma dei valori assoluti delle differenze di ogni linea dal numero magico.
- `correctLines`: La fitness di un individuo è data dall numero di linee magiche.

Ciascuna di queste funzioni di fitness ha una precisa espressione matematica in termini delle entrate della matrice e gode della proprietà che tutti quadrati magici di un certo hanno una fitness fitness definita (zero nei primi due casi e $2N+2$ nel terzo).

ii. Metodi di selezione e crossover

Ad ogni individuo è attribuita un valore di fitness, bisogna quindi farli riprodurli. Sono stati implementati diverti meccanismi per selezionare i genitori in modo da premiare contemporaneamente la bontà, ma anche per permettere sufficiente variabilità. Questi sono:

- `fitnessProportionate`: Ciascun individuo ha una probabilità di essere selezionato per fare crossover proporzionale alla sua fitness, o all'inverso della fitness per quelle funzioni per cui gli individui migliori sono quelli con fitness minore.

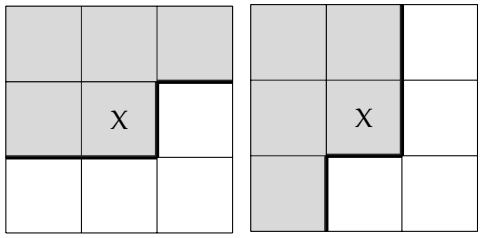
- `similarSquare`: Ciascun individuo ha una probabilità di essere selezionato per fare crossover proporzionale ad un fattore che dipende dalla fitness è dalla sua distanza dall'individuo migliore. In questo contesto per distanza si intende il numero di permutazioni necessarie a trasformare un quadrato in un altro. In qualche modo questo metodo è una implementazione di un sigma scaling [6].
- `fittests`: I migliori `nFittests` individui si riproducono con eguale probabilità, dopodiché la popolazione è ripristinata alla dimensione iniziale aggiungendo il giusto numero di individui casualmente generati.
- `elitism`: Un certo numero di individui è selezionato per passare direttamente alla generazione successiva, i rimanenti si riproducono con uno dei metodi precedenti.

Una volta che si è prodotta la lista dei genitori si fanno accoppiare scambiandosi geni con il crossover con una certa probabilità di crossover P_C. Sono stati implementati due tipi di crossover:

- Ad un punto verticale od orizzontale: si seleziona una posizione all'intero del quadrato, i figli sono ottenuti incrociando le porzioni di quadrato che precedono e che succedono tale posizione tra i due genitori.
- Ad due punti verticale od orizzontale: si selezionano due posizioni all'interno del quadrato: i figli sono ottenuti scambiando tra i due genitori i numeri compresi tra le due posizioni.

La distinzione tra verticale e orizzontale consiste nel criterio con cui alcuni numeri sono considerati antecedenti o posteriori alla posizione di scambio. In figura 3 è mostrata questa distinzione. Crossover verticale e orizzontale sono equivalenti a meno di una trasposizione.

Qualora sia in uso la fitness `correctLines` si utilizza invece un altro crossover: si selezionano una riga o una colonna da entrambi i genitori: i figli sono uguali ai genitori a meno di queste due linee scambiate.



(a) *Crossover orizziontale* **(b)** *Crossover verticale*

Figura 3: *Crossover ad un punto. Assumendo che sia estratto il punto X come punto di taglio per il crossover, in un caso si considera come elementi precedenti quelli che sono*

Siccome si vogliono costruire quadrati magici normali è necessario che tutti i numeri siano distinti, per questo motivo i crossover possono produrre degli individui non validi. Qualora questo succeda i quadrati vengono aggiustati eliminando i numeri doppi e sostituendoli con valori accettabili estratti casualmente.

iii. Implementazione in Mathematica

Per lanciare il programma si utilizza il comando `run[]`. Per gestire le varie possibili configurazioni con cui si desidera lanciare l'algoritmo genetico il comando supporta delle opzioni.

III. Algoritmo evolutivo

Gli algoritmi genetici si sono rivelati inadatti a risolvere il problema della costruzione dei quadrati magici, come verrà approfondito successivamente. Per superarne i problemi si è mostrato conveniente:

1. Eliminare la fase di crossover e aumentare il numero di mutazioni effettuate sul singolo individuo.

2. Effettuare controlli sistematici quando l'algoritmo comincia ad essere in condizioni di stallo.

Un algoritmo evolutivo è quindi un algoritmo genetico in cui non si effettuano crossover, ma si utilizzano esclusivamente mutazioni,

rese tuttavia più sofisticate per sopperire alla mancanza dei crossover, i quali in generale permettono di giungere più velocemente alla soluzione.

L'algoritmo evolutivo implementato in questo lavoro è basato sull'idea di Xie e Kang [10] che tiene conto anche della seconda miglioria.

i. Algoritmo di Xie-Kang

L'algoritmo di Xie-Kange oltre ad implementare i miglioramenti esposti all'inizio di questa sezione aggiunge un ulteriore contributo fondamentale: *la congettura della costruzione a due fasi*.

Per enunciare questa congettura è necessario definire una nuova tipologia di quadrati: i quadrati semimagici. Una matrice composta da numeri naturali differenti $N \times N$ è detta *quadrato semimagico* se è un quadrato magico a meno delle diagonali, ovvero se la somma dei valori su tutte le righe e su tutte le colonne è uguale al numero magico. Un quadrato semimagico è normale se le sue entrate sono tutti i numeri da 1 a N^2.

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.

Non è disponibile una dimostrazione per questa affermazione, che è per questo motivo definita una *congettura*.

i.1 Codifica degli individui

A differenza degli algoritmi genetici, nell'algoritmo evolutivo l'individuo non è direttamente il quadrato, bensì è una coppia di due matrici (M, Σ). Il primo quadrato, M, è quello che si cerca di rendere magico, mentre il secondo, Σ, contiene delle informazioni utili per le mutazioni. Gli elementi σ_{ij} di Σ determinano il range di valori che l'elemento m_{ij} può assumere a causa di una mutazione. Σ è una matrice dinamica, che subisce essa stessa mutazioni come M e che evolve tenendo in considerazione quanto M si discosta dall'essere un quadrato magico in termini di fitness e di linee non magiche.

La funzione di fitness f adottata è differente se il quadrato è semimagico oppure se possiede ancora linee non magiche:

- Per quadrati non semimagici:

$$f(M) = \sum_{i=1}^{N} \left| \sum_{j=1}^{N} a_{ij} - m \right| + \sum_{i=1}^{N} \left| \sum_{j=1}^{N} a_{ji} - m \right|$$

- Per quadrati semimagici:

$$f(M) = \left| \sum_{j=1}^{N} a_{jj} - m \right| + \left| \sum_{j=1}^{N} a_{j,N-j+1} - m \right|$$

Si definiscono $\mathrm{col}(k)$ come la somma di tutti gli elementi sulla colonna k e $\mathrm{row}(h)$ come la somma di tutti gli elementi sulla riga h. (Se una riga s è magica allora vale che $\mathrm{row}(s) = m$ e analogamente per le colonne). E similmente siano dg1 la somma dei termini sulla diagonale principale, e dg2 la somma dei termini sull'antidiagonale, cioè:

$$\mathrm{dg1} = \sum_{k=1}^{N} a_{kk} \qquad \mathrm{dg2} = \sum_{k=1}^{N} a_{k,-k}$$

Dove con $-k$ si intende l'indice $N - k + 1$, che è il k-esimo elemento della matrice leggendo da destra a sinistra. Con queste notazioni la funzione di fitness può essere riscritta come:

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

Si è scelto di utilizzare una fitness negativa per i quadrati semimagici in modo che ogni quadrato semimagico sia evolutivamente preferito a tutti i quadrati che hanno ancora linee non magiche. Questo accorgimento permette di trovare sempre l'individuo migliore di una popolazione anche se questa è composta da quadrati semimagici e non cercando l'individuo che ha fitness negativa più vicina a zero.

ii. Mutazioni

Siccome negli algoritmi evolutivi non vi è crossover, per esplorare lo spazio delle soluzioni

si utilizzano esclusivamente mutazioni e non è necessario lavorare con una popolazione composta da numero elevato di individui, ma se ne utilizza uno solo, il quale produce un numero fissato di figli che subiscono diverse mutazioni.

Sono state utilizzati due tipi di mutazioni: le mutazioni puntuali che coinvolgono solo due numeri, e le mutazioni lineari, le quali invece coinvolgono tutti i valori di una linea.

Sia randint(a, b) una funzione che estrae un numero intero nell'intervallo (a, b) con probabilità uniforme. Questa funzione è fornita da Mathematica come `Random[Integer]` e `RandomInteger[]`. Siano inoltre n_{row} e n_{col} rispettivamente il numero di righe e di colonne non magiche del quadrato.

ii.1 Mutazioni puntuali

Si definiscono degli insiemi di mutazione:

- S_1: Numeri la cui riga e colonna non è magica.
- S_{2r}: Numeri in righe non è magiche.
- S_{2c}: Numeri in colonne non è magiche.
- $S_2 = S_{2r} \cup S_{2c}$

E' evidente inoltre che:

$$S_1 = S_{2r} \cap S_{2c}$$

A meno che uno di questi insiemi non sia vuoto, ogni individuo è mutato con una delle seguenti tipologie di mutazione selezionata in modo casuale:

Mutazione da S_1 a S_2 Ogni elemento di S_1 è sottoposto a mutazione con probabilità $1/(n_{row}n_{col})$. Se un elemento x aventi indici i e j è selezionato per essere mutato, allora si calcola il valore:

$$n = x + \text{randint}(-\sigma_{ij}, \sigma_{ij})$$

Il quale viene aggiustato se produce risultati non accettabili:

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

Si cerca quindi in S_2 l'elemento che più si avvicina a n, cioè il numero t in S_2 tale che sia minimizzato il valore di

$$\min_{t \in S_2} |n - t|$$

Gli elementi n e t sono quindi scambiati. Una volta avvenuto lo scambio si effettua una mutazione anche sul valore di σ_{ij}. Il nuovo valore z di σ_{ij} è dato da:

$$z = \sigma_{ij} + \text{randint}(-1, 1)$$

Anche in questo caso per evitare di avere dei valori insensati si pone:

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

Dove σ_t definisce in qualche modo un "parametro di salute" del quadrato: σ_t è definito così:

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

Quando σ_t è piccolo significa che il quadrato è vicino ad essere magico, e quindi bisogna fare piccole mutazioni, mentre al contrario se σ_t è grande allora significa la fitness è alta e sono necessarie grandi mutazioni per rendere il quadrato magico.

Mutazione da S_2 a S_2 Ogni elemento x di S_2 è sottoposto a mutazione con probabilità che dipende determinata dall'appartenenza agli insiemi S_{2r} e S_{2c}:

$$P_M(x) = \begin{cases} 1/n_{row} & \text{se} \quad x \in S_{2r} \\ 1/n_{col} & \text{se} \quad x \in S_{2c} \\ 1/(n_{row}n_{col}) & \text{se} \quad x \in S_{2r} \cap S_{2c} \end{cases}$$

Se x è sottoposto a mutazione si esegue esattamente quello che si è fatto per la mutazione esposta precedentemente.

Mutazione da S_2 a M Questa mutazione è uguale alla precedente, a meno del fatto che gli elementi con cui si scambiano quelli di S_2 sono in tutta la matrice M.

ii.2 Mutazioni lineari

Una volta che l'algoritmo trova almeno un quadrato avente tutte le righe e le colonne magiche, cioè un quadrato semimagico, per evitare di perdere questi risultati vengono utilizzate mutazioni lineari al posto di quelle puntuali. Queste consistono nello scambio di due righe o di due colonne scelta casualmente, in questo modo sicuramente l'individuo continua ad avere tutte le righe e tutte le colonne magiche.

Per ogni individuo si eseguono N mutazioni lineari con probabilità unitaria. Si nota tuttavia che è possibile che le due linee che vengono estratte per essere scambiate coincidano, e ciò corrisponde a non aver effettuato alcuna mutazione. Equivalentemente si può quindi dire che la probabilità di mutazione lineare è di $1 - \frac{1}{N}$ considerando tuttavia solo linee differenti.

iii. Metodi di selezione

Una volta che tutta la popolazione è andata incontro a mutazione bisogna selezionare quale sarà il nuovo genitore. Vengono utilizzati due metodi di selezione (*evolution strategies*) differenti per incentivare o meno la variabilità degli individui. Questi sono metodi standard negli algoritmi evolutivi:

- $(\mu, \lambda) - ES$: La nuova generazione di genitori è formata dai migliori figli della precedente.
- $(\mu + \lambda) - ES$: La nuova generazione è formata dagli individui migliori tra i genitori e i figli della generazione precedente.

In queste diciture μ indica i genitori della generazione precedente, mentre λ i figli. L'utilizzo della seconda strategia evolutiva garantisce che non vengono persi individui buoni ma modificati dalle mutazioni, mentre l'utilizzo del primo permette di esplorare più velocemente lo spazio delle soluzioni.

Nell'algoritmo implementato in questo lavoro sono state utilizzate entrambe le strategie evolutive a seconda di quale sia più utile nel momento della selezione.

iv. Rettificazioni locali

Senza intervenire direttamente in modo sistematico sulla costruzione dei quadrati, anche l'algoritmo evolutivo, come quello genetico, non è molto più efficiente di una ricerca casuale nello spazio delle soluzioni, e per questo motivo non converge quando lo spazio delle soluzioni è troppo grosso, e fallisse già con quadrati di piccole dimensioni. Per ovviare a questo problema Xie e Kang hanno proposto di utilizzare delle *rettificazioni locali*: quando gli individui sono abbastanza buoni vengono passati in rassegna per cercare se vi sono degli scambi che diminuiscono il numero di righe sbagliate oppure che rendono le diagonali magiche. Si definiscono *rettificazioni locali lineari* quelle che hanno come obiettivo la diminuzione delle righe non magiche, mentre *rettificazioni locali diagonali* quelle che intendono aggiustare le diagonali.

iv.1 Rettificazioni locali lineari

Sono state implementate due tipologie di rettificazioni locali lineari: lo scambio di una coppia, e lo scambio di due coppie tali che soddisfino alcune condizioni. Queste condizioni garantiscono che lo scambio renda magica una o più linee senza rovinare le altre. Non si fanno analisi a più di due coppie perché sarebbe un compito computazionalmente troppo oneroso. Ricordando che $\mathrm{col}(k)$ è la somma degli elementi sulla $k-$esima colonna e $\mathrm{row}(h)$ è la somma degli elementi sulla $h-$esima riga:

Scambio di una coppia Due numeri a_{ks} e a_{ls} sono scambiati alla riga k e l e alla colonna s se sono soddisfatte:

- $\mathrm{row}(k) - m = a_{ks} - a_{ls}$
- $m - \mathrm{row}(l) = a_{ks} - a_{ls}$

Un esempio di questa rettificazione è riportato in appendice.

Scambio di due coppie I numeri a_{ks} e a_{lt} sono scambiati con i numeri a_{ls} e a_{kt} (il primo

con il primo e il secondo con il secondo), corrispondenti alle righe k e l e alle colonne s e t se sono soddisfatte:

- $\text{row}(k) - m = a_{ks} + a_{kt} - a_{lt} - a_{ls}$
- $m - \text{row}(l) = a_{ks} + a_{kt} - a_{lt} - a_{ls}$

Le condizioni duali sono applicate anche alle colonne con le ovvie sostituzioni.

iv.2 Rettificazioni locali diagonali

Una volta che si è realizzato un quadrato semimagico per ottenere il quadrato magico bisogna intervenire utilizzando metodi deterministici. I metodi implementati sono scambi di coppie di elementi, oppure di intere linee. Ricordando che dg1 e dg2 sono la somma degli elementi sulla diagonale e sull'antidiagonale rispettivamente.

Scambio puntuale Se sono soddisfatte le condizioni:

- $a_{ii} + a_{ij} = a_{ji} + a_{jj}$
- $\left(a_{ii} + a_{jj}\right) - \left(a_{ij} + a_{ji}\right) = \text{dg1} - m$

allora a_{ii} è scambiato con a_{ji} e a_{ji} con a_{jj}. In modo simmetrico per l'antidiagonale.

Scambio lineare Se sono soddisfatte le condizioni:

- $\left(a_{ii} + a_{jj}\right) - \left(a_{ij} + a_{ji}\right) = \text{dg1} - m$
- $\left(a_{i,-i} + a_{j,-j}\right) - \left(a_{i,-j} + a_{j,-i}\right) = \text{dg2} - m$

allora la riga i è scambiata con la riga j. Simmetricamente con colonne.

v. Implementazione in Mathematica

Il punto critico dell'implementazione sono le rettificazioni locali: è possibile implementarle senza manipolare direttamente gli indici e quindi sfruttando gli strumenti nativi di Mathematica?

Quando si esegue questa correzione tutto il quadrato viene passato in rassegna, per far ciò si utilizzano cicli innestati: uno che fissa il primo elemento, e il secondo che fa scorrere l'intero quadrato in cerca di numeri che soddisfino le condizioni richieste. Per questo motivo in generale il numero di cicli innestati è quattro, due per gli indici del primo elemento, e due per gli indici del secondo elemento, per questo motivo la complessità della rettificazione è sicuramente maggiore a $\mathcal{O}(N^4)$.

L'itero algoritmo è esposto nel diagramma di flusso riportato in figura 7

IV. Risultati

i. Algoritmi genetici

Gli algoritmi genetici implementati non sono stati in grado di produrre i risultati sperati: nessuna combinazione di fitness, e criterio di selezione è risuscita a produrre quadrati di dimensioni superiori al 3×3. I risultati ottenuti sono riportati in tabella 2.

Tabella 2: *Risultati ottenuti: N è l'ordine del quadrato, F, C la funzione di fitness e il criterio di selezione, n_{tent} il numero di esecuzioni dell'algoritmo, n_{ok} il numero di quadrati costruiti con successo entro 10 000 generazioni. Le abbreviazioni utilizzate sono riportate in tabella 3*

N	F,C	n_{tent}	n_{ok}
3	tS, fP, E	10	7
3	tS, fP	10	4
3	tA, fP, E	10	8
3	cL, F, E	10	6
3	tS, sS, E	10	1
4	tS, fP, E	10	0
4	tS, F, E	10	0
4	tA, fP	10	0
4	tA, sS	10	0
4	cL, fP	10	0
10	cL, fP, E	10	0
10	tS, fP, E	10	0
10	tS, F, E	10	0

ii. Algoritmo di Xie-Kang

Al contrario degli algoritmi genetici, l'algoritmo evolutivo ha prodotto con successo un discreto numero di quadrati magici. Sono stati

Tabella 3: *Abbreviazioni utilizzate nella tablela 2.*

Sigla	Metodo
fP	`fitnessProportionate`
sS	`similarSquare`
F	`fittests`
E	`elitism`
tS	`totalSquared`
tA	`totalAbs`
cL	`correctLines`

costruiti quadrati di ordine dal 10 a 40, e in nessun caso l'algoritmo si è trovato in una condizione metastabile e non è giunto a convergenza. Curiosamente, invece, per quadrati di dimensioni piccole (da 4×4 a 6×6) è capitato che il programma rimanga in una condizione metastabile.

Si sono raccolte delle statistiche per caratterizzare come varia il tempo di esecuzione del programma al variare della dimensione dell'input e per confrontare i dati raccolti con quelli di Xie e Kang.

Nella tabella 4 sono riportati i dati raccolti, mentre nella figura 4 sono rappresentati i tempi di esecuzione al variare dell'ordine del quadrato con un possibile andamento interpolante. I dati raccolti si accordato con una legge di potenza avente per esponente un valore compreso tra 5 e 6. Con i dati a disposizione questo esponente vale ~ 5.26 con un coefficiente di determinazione $R^2 = 0.96$. Questo è un miglioramento sostanziale rispetto alla ricerca *brute-force*.

I fit sono realizzati con il metodo dei minimi quadrati.

Confrontando i dati ottenuti con quelli di Xie e Kang si nota che l'algoritmo implementato in questo lavoro si comporta decisamente peggio in termini di legge di scala rispetto all'originale. Si nota tuttavia che il numero di generazioni necessarie per raggiungere il quadrato magico è sempre notevolmente minore (circa inferiore del 50 %) rispetto a quelle richieste di Xie e Kang: questo è dovuto al fatto che in questa implementazione ogni generazione è composta da una popolazione di 25 individui, mentre nell'originale di Xie e Kang sono solo 10. Si è

Tabella 4: *Risultati ottenuti: N è l'ordine del quadrato, n_{tent} il numero di esecuzioni dell'algoritmo, n_{ok} il numero di quadrati costruiti con successo entro 30 000 generazioni, τ il tempo medio di costruzione, τ/τ_0 è il rapporto tra il tempo medio con il tempo medio per la costruzione del quadrato con $N = 10$, n_{gen} è il numero medio di generazioni.*

N	n_{tent}	n_{ok}	τ	τ/τ_0	n_{gen}
3	10	10	0.12 s	0	3
10	10	10	55 s	1	178
15	10	10	5.75 min	6	536
20	10	10	31.2 min	33	1085
25	10	10	1.73 h	112	1893
30	10	10	4.23 h	275	3007
35	10	10	12.38 h	824	4359
40	10	10		1	

Tabella 5: *Risultati ottenuti: N è l'ordine del quadrato, n_{tent} il numero di esecuzioni dell'algoritmo, n_{ok} il numero di quadrati costruiti con successo entro 30 000 generazioni, τ il tempo medio di costruzione, τ/τ_0 è il rapporto tra il tempo medio con il tempo medio per la costruzione del quadrato con $N = 10$, n_{gen} è il numero medio di generazioni.*

N	n_{tent}	n_{ok}	τ	τ/τ_0	n_{gen}
3	10	10	0.22 s	0	4
10	10	10	69 s	1	377
15	10	10	4.37 min	3	1109
20	10	10	26.3 min	22	2852
25	10	10	1.51 h	78	4855
30	10	10	4.23 h	275	3007
35	10	10	12.38 h	824	4359
40	10	10		1	

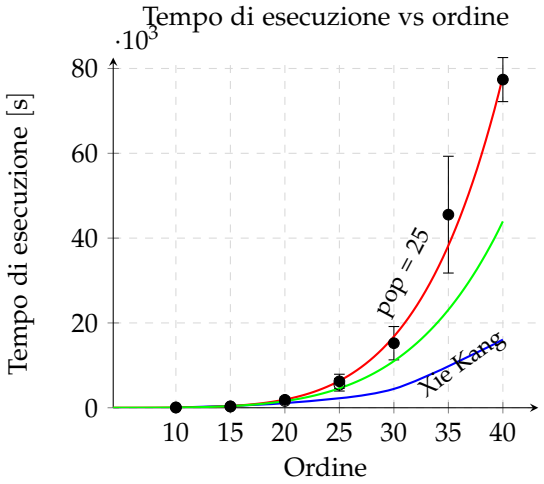


Figura 4: *Tempi di computazione medi su dieci esecuzioni dell'algoritmo evolutivo per quadrati di ordine dal 10 al 30. La curva rossa descrive una possibile legge di scala: è una legge di potenza αx^β con esponente β compreso tra 5 e 6. Le barre di errore sono le deviazioni standard delle distribuzioni dei dieci risultati effettuati per ogni ordine. In blu sono i dati di Xie e Kang, rinormalizzati sul tempo di computazione. I fit sono realizzati con il metodo dei minimi quadrati.*

adoperata questa scelta perché si è verificato che per i quadrati 10×10 la convergenza avviene circa il 20 % più velocemente. E' tuttavia possibile che per ordini superiori questa scelta sia dannosa e abbia causato questo bias nei tempi.

La dipendenza della dimensione della popolazione inoltre incide anche per un ulteriore motivo: le mutazioni e le rettificazioni sono eseguite in parallelo su tutti i core a disposizione. Nella macchina usata venivano in effetti istanziati quattro thread, e per questo motivo le popolazioni che non sono potenze di 4 non sono ottimali.

Inoltre si nota che Xie e Kang hanno trovato che la legge di scala è quasi perfettamente come N^4 (con i dati a disposizione il corretto esponente di una power law interpolante è 4.04), il quale è il limite inferiore ottenuto da considerazioni teoriche nella implementazione realizzata in questo lavoro.

V. Conclusioni

i. Limiti degli algoritmi stocastici

In questo lavoro sono stati analizzati due possibili algoritmi stocastici per la costruzioni di quadrati magici, di cui, tuttavia, solo uno è in grado di arrivare effettivamente ad un risultato.

Il crossover risulta inutile in quanto la prescrizione che il quadrato abbia tutti numeri diversi impedisce al crossover di produrre un individuo migliore a partire da due quadrati buoni selezionando opportunamente le righe. A causa delle ricostruzioni del quadrato dopo il crossover, gli algoritmi genetici con crossover per questo scopo sono quasi equivalenti alla ricerca casuale.

Il motivo per cui gli algoritmi genetici non convergono è che non si può trovare una funzione di fitness che individui un un percorso verso l'individuo perfetto. E' possibile costruire un quadrato che è perfetto a meno di due sole permutazioni ma che ha fino a otto linee non magiche, e viceversa è possibile costruire un quadrato che è ha tutte le linee magiche

a meno di due, eppure che sia molto distante in termini di permutazioni da un quadrato magico.

ii. Confronto con i risultati di Xie e Kang

L'attuale implementazione è quindi adatta per la costruzioni di quadrati di ordine relativamente piccolo.

Anche questa implementazione individua tre fasi nella ricerca del quadrato magico:

1. Durante la prima fase la fitness decresce vertiginosamente mentre il numero di linee non magiche rimase costante: nessuna linea ha la somma corretta.
2. Quando la fitness è sufficientemente bassa (inferiore a $50 \times N$) cominciano ad essere utilizzati i meccanismi di rettificazione locale lineare: questo aggiusta molte linee, ma non abbassa sensibilmente la funzione di fitness.
3. Infine si riesce a produrre un quadrato semimagico, in questa fase si utilizzano le permutazioni lineari e le rettificazioni per esplorare tutto lo spazio delle soluzioni, come mostra il grafico 6a, in cui le brusche oscillazioni di fitness indicano che molte soluzioni differenti sono state tentata.

L'indagine sui tempi di esecuzione delle varie subroutine dell'algoritmo mostra come la quasi totalità del tempo sia spesa nel cercare di rettificare le linee del quadrato, in particolare nella rettificazione a due coppie. Ci si attendeva questo comportamento in quanto l'implementazione ha richiesto l'utilizzo di quattro cicli innestati per controllare tutte le coppie di elementi del quadrato. Questo significa che per un quadro $N \times N$ in questa fase vengono eseguiti circa N^4, che

Le rettificazioni ad una coppia invece effettuano $2(N-1)N^2 \sim N^3$ controlli ogni volta, e quindi per quadrati grandi influiscono sensibilmente meno sul tempo di elaborazione. Per questo non si sono implementate rettificazioni che coinvolgono un numero maggiore di due di coppie.

Tabella 6: *Tempi di esecuzioni medi delle varie routine dell'algoritmo per quadrato di ordine 20. Siccome* `rectifyLines` *ha complessità maggiore degli altri metodi, quindi aumentando l'ordine tende a occupare tutto il tempo di elaborazione. Il restante tempo è speso nell'eseguire operazioni per produrre le statistiche. L'ultima riga specifica come si suddivide il tempo di esecuzione della routine* `rectifyLines` *nelle due subroutine per la rettificazione a una coppia o a due coppie. Questi dati sono nel caso della popolazione di 25 elementi.*

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

iii. Possibili sviluppi futuri

L'attuale implementazione dell'algoritmo evolutivo riesce sempre a produrre quadrati magici, tuttavia il tempo di elaborazione necessario per costruire quadrati di grandi dimensioni è molto elevato, come si nota dalla tabella 4, e per questo motivo è impensabile utilizzarlo per costruire quadrati ancora più grandi. La quasi totalità del tempo di elaborazione è spesa nel cercare di rettificare i quadrati, questo perché tali processi devono passare in rassegna l'intero quadrato più volte alla ricerca di determinate condizioni, e per questo motivo l'implementazione contiene numerosi cicli nested, che la rendono molto pesante dal punto di vista del tempo di esecuzione.

Così come è implementata questa funzione non può essere parallelizzata perché si cerca di eseguire tutti i possibili scambi che migliorano il quadrato, e una volta effettuato uno scambio il quadrato su cui si lavora cambia. Tuttavia è possibile implementare la funzione di rettificazione in modo che quando è eseguito uno scambio termini l'esecuzione, in questo modo sono necessarie più generazione, ma diventa possibile implementare la funzione in modo altamente parallelo: ad ogni thread è dato un sottoinsieme di tutte i controlli che bisogna ef-

fettuare, quando uno di questi risulta positivo la routine termina e si effettua lo scambio. In questo modo si evita che se in un thread la condizione è verificata e si effettua uno scambio gli altri thread lavorino con il quadrato nello stato precedente. Questa parallelizzazione porterebbe notevoli vantaggi in termini di tempo di computazione della singola generazione.

Il tempo di computazione può essere sensibilmente ridotto riscrivendo i metodi di rettificazione in modo da renderli compilabili con il comando `Compile[]`.

Mathematica si è rivelato non essenziale ai fini dell'implementazione dell'algoritmo, e anzi probabilmente ne ha limitato le potenzialità. Alcune funzioni native, come `Map[]`, `Table[]`, `Replace[]` sono state utilizzate pesante e si sono rivelate strumenti di grande aiuto per la semplificazione del codice. La gestione delle liste di Mathematica ha permesso di scrivere in maniera concisa ed elegante tutti i metodi che si applicano alla intere popolazioni intere.

Tuttavia, il cuore dell'algoritmo, cioè i metodi di rettificazione, richiedono che si percorra l'intero quadrato più volte lavorando esplicitamente con gli indici, e questo costituisce un grosso limite per `Mathematica`, il quale non gestisce in modo sufficientemente le risorse per questi cicli in maniera altrettanto efficiente rispetto a linguaggi compilati come il `C`.

Il fatto che sia possibile in linea teorica compilare l'intero algoritmo di Xie e Kang con Mathematica mostra come effettivamente non sia necessario tale linguaggio ai fini dell'implementazione.

Si è mostrato come l'approccio stocastico + deterministico sia re

Eseguire dieci costruzioni di quadrati dal 3×3 al 40×40 richiede circa due settimane di computazione, per questo motivo non si è ulteriormente indagata come la dimensione della popolazione, che è sostanzialmente l'unico parametro che è possibile impostare direttamente, incida sul tempo di esecuzione. Si è tuttavia mostrato che questo parametro ha un impatto non trascurabile: si potrebbe rendere ulteriormente l'algoritmo più veloce trovando il parametro ottimale.

VI. Appendice

i. Calcolo del numero magico per un quadrato magico normale

Dato un quadrato magico normale (cioè che contiene tutti i numeri da 1 a N) allora ciascuna linea e le diagonali hanno per somma il valore magico m, questo significa che sommando tutti i numeri del quadrato si ottiene Nm che è la somma di tutte le somme di ogni riga. Quindi:

$$Nm = \sum_{n=1}^{N^2} n$$

Ma la somma al membro destro può essere valutata esplicitamente:

$$\sum_{n=1}^{N^2} n = \frac{1}{2} N^2 (N^2 + 1)$$

Da cui utilizzando la relazione precedente:

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

ii. Esempio di rettificazione lineare

Si riporta qui un esempio di rettificazione locale lineare: con una permutazione si passa da un quadrato con una sola riga magica, ad uno con tutte le righe magiche.

1	5	6
4	3	8
2	7	9

(a) *Prima*

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

(b) *Dopo*

Figura 5: *Esempio di rettificazione lineare. Dopo la rettificazione la prima è l'ultima riga diventano magiche.*

Riferimenti bibliografici

[1] Abe, G. (1994). Unsolved Problems on Magic Squares. *Discrete Mathematics (1993) 127 (1–3).*

[2] Ahmed, K. e Ender Ö. (2014). Constructing Constrained-Version of Magic Squares Using Selection Hyper-heuristics. *The Computer Journal (2014) 57 (3)*.

[3] Boyer, C. `http://www.multimagie.com`.

[4] Kraitchik, M. (1942). Mathematical Recreations. Norton, New York.

[5] Loly, P. (2003). Scientific Studies of Magic Squares. `http://home.cc.umanitoba.ca/~loly/IHPST.pdf`.

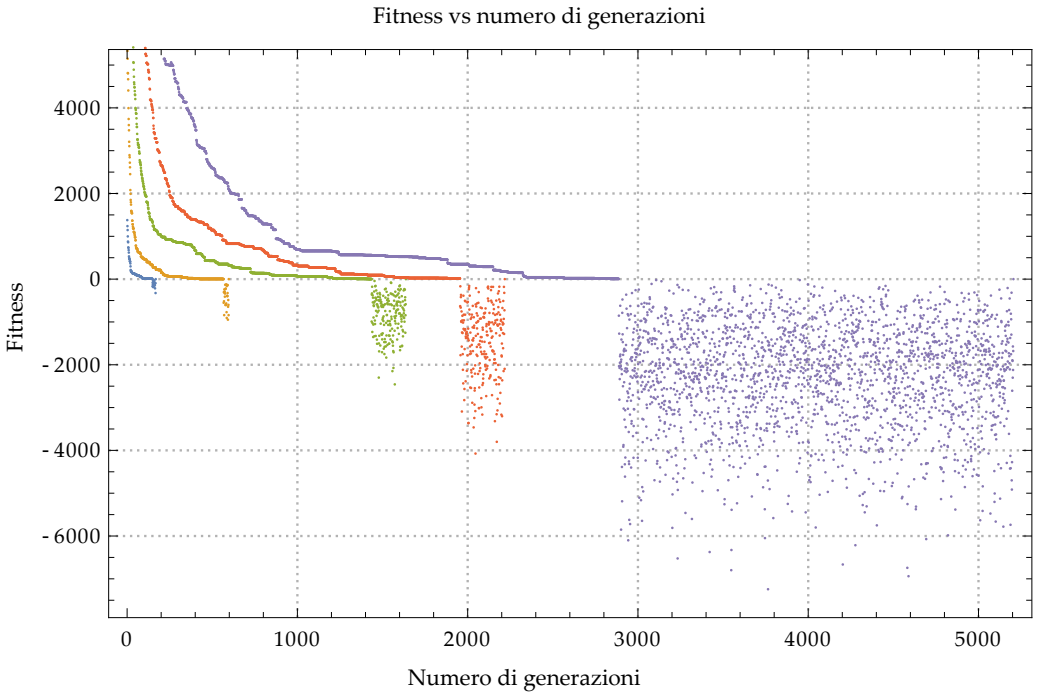
[6] Mitchell, M. (1996). An Introduction to Genetic Algorithms. MIT Press, Cambridge.

[7] Saha, B. e Bhattacharya, S (2015). An Approach To Hiding Image Into Video Using Magic Square *International Conference on Computer Science and Engineering, 2012*.

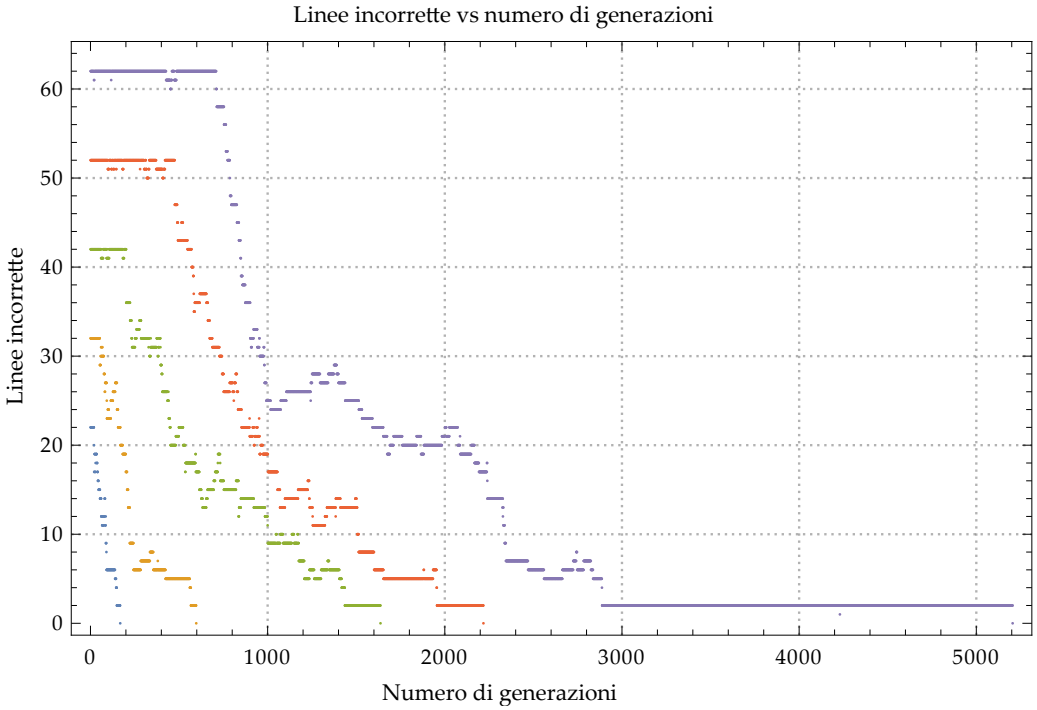
[8] Tomba, I. e Shibiraj, N (2014). Successful Implementation of the Hill and Magic Square Ciphers: A New Direction. *International Journal of Advanced Computer Technology* .

[9] Trump, W. (2015). `http://www.trump.de/magic-squares/estimates/index.html`.

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



(a) *Fitness vs generazione al variare dell'ordine dei quadrati. Dove la fitness è negativa il quadrato è semimagico è si fanno rettificazioni locali diagonali. In questa fase non è rappresentato il nuovo genitore, ma il figlio migliore. Questo mostra come sia esplorato lo spazio delle soluzioni.*



(b) *Linee non magiche vs generazione al variare dell'ordine dei quadrati.*

Figura 6: *Statistiche al variare dell'ordine del quadrato: blu 10, giallo 15, verde 20, rosso 25, viola 30.*

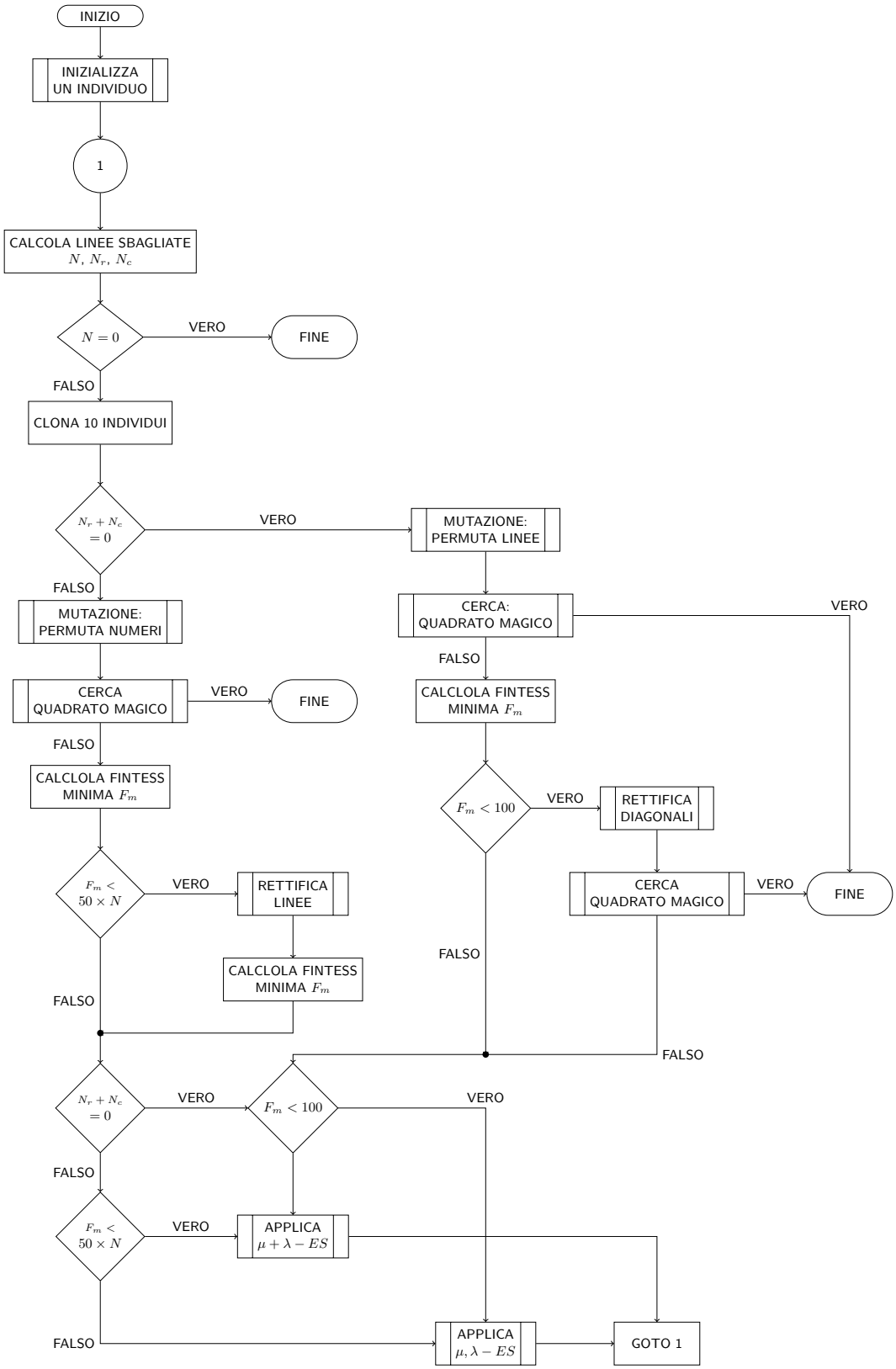


Figura 7: *Diagramma di flusso dell'algoritmo di Xie e Kang.*