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, ovvero matrici di numeri naturali tutti differenti in cui la somma dei valori su ogni riga, su ogni colonna e sulle diagonali è sempre la stessa. La prima implementazione è un algoritmo genetico realizzato utilizzando diverse funzioni di fitness e metodi di selezione, mentre il secondo è un algoritmo evolutivo, basato sul lavoro di Xie e Kang [1]. 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 adatto il problema. L'algoritmo evolutivo invece 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 parziale accordo.

I. Introduzione

Un quadrato magico di ordine N è una matrice quadrata M di dimensioni $N \times N$ contenente numeri naturali distinti tale che la somma dei valori su ciascuna riga, colonna e diagonale sia sempre la stessa, detta *numero magico*. Qualora i numeri che compaiono sono i primi N^2 allora si il quadrato è detto *normale*, e la somma che devono avere i termini sulla stessa linea è detta *numero o costante magica*, e può essere calcolata.

Si trova che la costante magica di un quadrato magico di ordine N è:

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

Chinese literature dating from as early as 2800 B.C., when a Magic Square known as the "Loh-Shu", or "scroll of the river Loh" (see above), was invented by Fuh-Hi, the mythical founder of Chinese civilisation Greek writings dating from about 1300 B.C. the works of Theon of Smyrna in 130 A.D. use by Arabian astrologers in the ninth century when drawing up horoscopes Arabic literature, written by Abraham ben Ezra, dating from the eleventh century India, dating from the eleventh or twelfth century, where the earliest fourth order magic square was found, in Khajuraho the writings of the Greek mathematician, Emanuel Moschopulus, whose works now reside in the National Library in Paris more recently, magic squares appeared in Chinese literature during the latter part of the posterior Chou dynasty (951 - 1126 A.D.) or the beginning of the Southern Sung dynasty (1127 - 1333 A.D.) the works of Cornelius Agrippa, a German physician and theologian from the sixteenth century, who constructed seven magic squares, of orders three to nine inclusive, which he associated with the seven planets then known (including both the Sun and the Moon) art, with the relatively well-known magic square which can be found in Albert Drer's engraving " Melencolia", where the date of its creation, 1514 AD, may be seen in the centre two cells of the bottom row a detailed French work, published in 1838 A.D.

Gli algoritmi sono quindi implementati in Mathematica 8. Si è utilizzata questa versione

Tabella 1: *Numero di quadrati magici. Dati di [6].*

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

perché permette di parallelizzare alcune funzioni in modo estremamente semplice, come ad esempio con il comando `Paralellize[]`.

In questo lavoro si utilizzerà il termine *linea* indicando in modo generico una riga o una colonna.

Una linea si dice *magica* se la somma dei numeri che la compongono è il numero magico.

Trovare quadrati magici è un compito difficile in quanto il loro numero è molto piccolo rispetto a tutte le possibilità (non esiste ancora una formula che permetta di calcolare il numero di quadrati magici di ordine N, le attuali stime sono ottenute utilizzando metodi Monte Carlo e approcci con tecniche di meccanica statistica).

I quadrati magici hanno anche applicazioni tecnologiche, tra cui nella crittografia [2], nella steganografia [3] (la tecnica di occultare informazioni nelle immagini), ma anche in teoria dei grafi o dei giochi, e in molti altri campi, anche in fisica

i. Approcci deterministici

Metodi per costruire quadrati magici sono disponibili già da molti anni.

Alcuni semplici metodi generali di costruzione sono quelli elencati da Kraitchik [5]

ii. Approcci stocastici

Nonostante gli algoritmi elencati nella sezione precedente siano computazionalmente molto efficienti nella costruzione di quadrati magici, questi hanno una limitazione: fissato l'ordine generano sempre il medesimo quadrato. In questo modo questi 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).

II. Algoritmi genetici

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.

i. Funzioni di fitness

Le funzioni di fitness che sono state implementate sono:

- `totalSquared`:

$$f(M) = \{\text{Somma dei quadrati delle deviazioni di ogni linea dal numero magico}\}$$

- `totalAbs`:

$$f(M) = \{\text{Valore assoluto delle differenze di ogni linea dal numero magico}\}$$

- `correctLines`:

$$f(M) = \{\text{Numero di linee magiche}\}$$

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

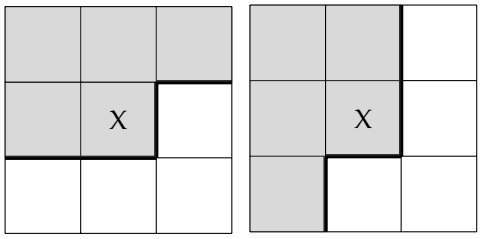
Metodi di selezione:

- fitnessProportionate
- similarSquare
- fittests
- elitism

Metodi di crossover:

- Ad un punto verticale od orizzontale
- Ad due punti verticale od orizzontale

Crossover verticale e orizzontale sono equivalenti a meno di una trasposizione.



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

Figura 1: *Crossover ad un punto*

iii. Implementazione in Mathematica

iv. Conclusioni

III. Algoritmo evolutivo

Per superare i problemi legati all'applicazione di un algoritmo genetico per la costruzione di quadrati magici normali si adottano due accorgimenti:

1. Si elimina la fase di crossover e si aumenta il numero di mutazioni effettuate sul singolo individuo.

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

Siccome ora non vi sono più crossover 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.

In questo lavoro è stata implementato un'algoritmo basato su quello proposto da Xie e Kang in [1].

i. Algoritmo di Xie-Kang

Tale algoritmo implementa i due miglioramenti esposti all'inizio di questa sezione e aggiunge un ulteriore contributo fondamentale: *la congettura della costruzione a due fasi*.

i.1 Congettura della costruzione a due fasi

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 ddei 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.

La congettura della costruzione a due fasi di Xie e Kang afferma che 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.

ii. Metodi di selezione

Vengono utilizzati due metodi di selezione (*evolution strategies*) differenti per incentivare o meno la variabilità degli individui.

- $(\mu, \lambda) - ES$: 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.

In questo algoritmo sono stati utilizzate entrambe le strategie evolutive a seconda di quale sia più utile nel momento della selezione. Inoltre siccome non vi è un reale vantaggio nell'avere un numero elevato di genitori se ne seleziona sempre uno solo, il quale produce 25 figli.

iii. Mutazioni

iii.1 Mutazioni puntuali

Nella prima fase dell'algoritmo ogni individuo è mutato con una delle seguenti tre mutazioni casualmente selezionata.

Si definiscono tre insiemi di mutazione:

$$S_1 = \{\text{Numeri la cui riga e colonna non è magica}\}$$

$$S_{2r} = \{\text{Numeri in righe non è magiche}\}$$

$$S_{2c} = \{\text{Numeri in colonne non è magiche}\}$$

$$S_2 = S_{2r} \cup S_{2c}$$

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 si calcola il valore:

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

Inoltre per evitare che n sia un numero non accettabile:

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

Si cerca quindi in S_2 l'elemento che più si avvicina a questo. Ciò il numero t in S_2 tale che sia minimizzato $\min_{t \in S_2} |n - t|$. Questi due elementi vengono quindi scambiati.

Una volta che si sono scambiati i due numeri si effettua una mutazione anche sul valore di σ. Il nuovo valore di σ_{ij} è dato da:

$$n = x + \text{randint}(-1, 1)$$

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

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

Mutazione da S_2 a S_2 Ogni elemento di S_2 è sottoposto a mutazione con probabilità che dipende dal fatto che se è appartiene a S

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.

iii.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 estratte 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 dire che la probabilità di mutazione lineare è di $1 - \frac{1}{N}$ considerando tuttavia solo linee differenti.

iv. Rettificazioni locali

Senza intervenire direttamente in modo sistematico sulla costruzione dei quadrati, anche

Tabella 2: *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 30000 generazioni, τ il tempo medio di costruzione in secondi, 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 (s)	τ/τ_0	n_{gen}
10	10	10		1	
15	10	10		1	
20	10	10		1	
25	10	10		1	
30	10	10		1	
35	10	10		1	
40	10	10		1	
45	10	10		1	
50	10	10		1	

l'algoritmo evolutivo, come quello genetico non è molto più efficiente di una ricerca casuale tra lo spazio delle soluzioni, e per questo motivo non converge quando lo spazio delle soluzioni è troppo grosso, quindi già con quadrati 6×6

iv.1 Rettificazioni locali lineari

iv.2 Rettificazioni locali diagonali

v. Implementazione in Mathematica

IV. Risultati

i. Algoritmi genetici

ii. Algoritmo di Xie-Kang

Tabella 3: *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 10000 generazioni.*

N	F, C	n_{tent}	n_{ok}
3		10	0
4		10	0
4		10	0
4		10	0
4		10	0
35		10	0
40		10	0
45		10	0
50		10	0

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 dlele ricostruzione 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

V. Conclusioni

i. Limitazioni 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.

ii. Confronto con i risultati di Xie e Kang

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

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 2, 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

Tabella 4: *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.*

Routine	Tempo speso (%)
`mutate`	
`selectFittest`	
`rectifyLines`	
`rectifyDiagonals`	

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.

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. 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.

VI. Appendice

Esempio rettificazione su riga 1:

Prima:

1	5	6
4	3	8
2	7	9

Dopo:

1	5	9
4	3	8
2	7	6

Esempio di rettificazione diagonale 2:

9	5	1
3	4	2
8	7	6

(a) *Prima*

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

(b) *Dopo*

Figura 2: *Esempio di rettificazione diagonale 2*

Dopo:

Righe e colonne magiche

9	5	1
3	4	2
8	7	6

9	1	5
2	6	7
4	8	3

Riferimenti bibliografici

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

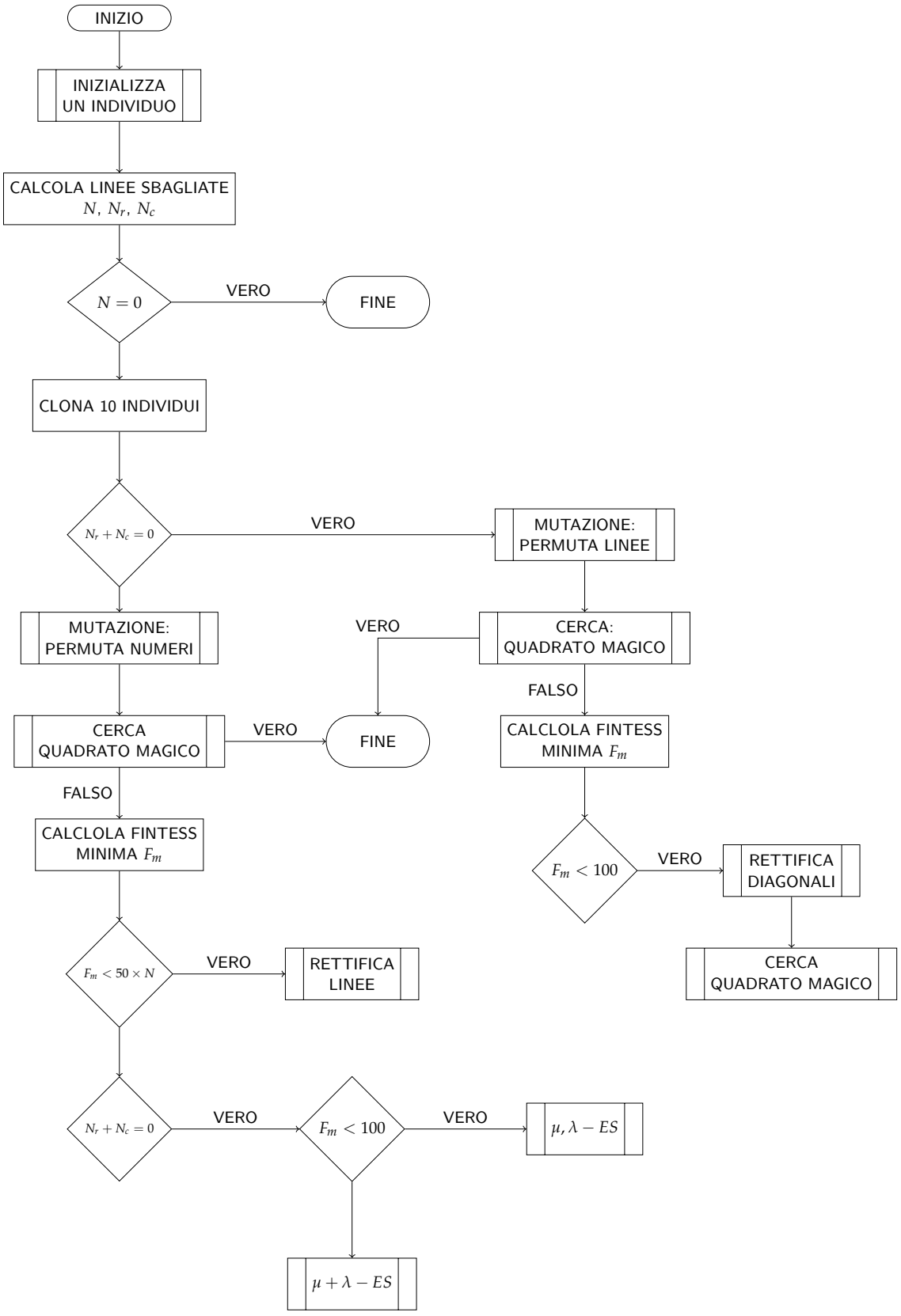
[2] Tomba, I. e Shibiraj, N (2014). Successful Implementation of the Hill and Magic Square Ciphers: A New Direction. *International Journal of Advanced Computer Technology* .

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

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

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

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


INIZIO
INIZIALIZZA UN INDIVIDUO
CALCOLA LINEE SBAGLIATE N, N_r, N_c
$N = 0$
VERO
FINE
CLONA 10 INDIVIDUI
$N_r + N_c = 0$
VERO
MUTAZIONE: PERMUTA LINEE
MUTAZIONE: PERMUTA NUMERI
VERO
CERCA: QUADRATO MAGICO
FALSO
CERCA QUADRATO MAGICO
VERO
FINE
CALCLOLA FINTESS MINIMA F_m
FALSO
CALCLOLA FINTESS MINIMA F_m
$F_m < 100$
VERO
RETTIFICA DIAGONALI
CERCA QUADRATO MAGICO
$F_m < 50 \times N$
VERO
RETTIFICA LINEE
$N_r + N_c = 0$
VERO
$F_m < 100$
VERO
$\mu, \lambda - ES$
$\mu + \lambda - ES$