Práctica 1
de
Computación Bio-Inspirada

Ana Gil Molina
Levi Malest Villarreal

06/03/2025

Índice

1. **Introducción al problema de las *n*-reinas**
2. **Introducción a los algoritmos evolutivos**
3. **Resultados con AE1**
4. **Resultados con AE2**
5. **Tests estadísticos**
6. **Conclusiones**
7. **Pruebas extra**

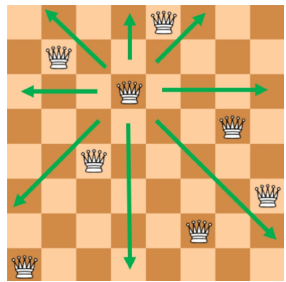
Introducción al problema de las n-reinas

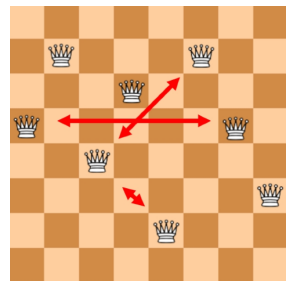
Introducción al problema de las n-reinas

Consiste en colocar *n* reinas en un tablero de ajedrez de forma que ninguna reina ataque a otra.

Dos reinas no pueden compartir:

- Misma fila
- Misma columna
- Misma diagonal





Introducción al problema de las n-reinas
Representación de los individuos

Representaremos cada individuo de la población mediante una permutación $x_i^t = (x_{i1}^t, x_{i2}^t, \ldots, x_{in}^t)$, donde:

- Cada índice representa una fila.
- El valor en cada posición representa la columna donde está la reina en esa fila.

Por ejemplo, para $n = 4$, la permutación $(4, 2, 1, 3)$ significa:

- Reina en fila 1, columna 4
- Reina en fila 2, columna 2
- Reina en fila 3, columna 1
- Reina en fila 4, columna 3

Introducción al problema de las n-reinas
Función fitness

La representación con permutaciones elimina los conflictos en las filas y las columnas.

Las diagonales todavía pueden tener conflictos. Dos reinas están en la misma diagonal si:

1. $fila_1 - columna_1 = fila_2 - columna_2$ (diagonal ↘)
2. $fila_1 + columna_1 = fila_2 + columna_2$ (diagonal ↗)

Esto es equivalente a la condición: $|fila_1 - fila_2| = |columna_1 - columna_2|$

Función fitness a minimizar:

$$f(x) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \delta(|x_i - x_j| = |i - j|)$$

Solución válida si $f(x) = 0$.

Introducción a los algoritmos evolutivos

Introducción a los algoritmos evolutivos

- **AE1:** Selección proporcional, muestreo estocástico con reemplazo, sustitución generacional completa.
- **AE2:** Selección por ranking exponencial ($c = 0{,}3$), muestreo estocástico con reemplazo, sustitución generacional completa.

Introducción a los algoritmos evolutivos
Selección

- La **selección proporcional** asigna a cada individuo la siguiente probabilidad:

$$p_i^t = \frac{f(x_i^t)}{\sum_{i=1}^{N} f(x_i^t)}, i = 1, \ldots, N$$

- La **selección por ranking exponencial** asigna a cada individuo la probabilidad:

$$p_i^t = \frac{c-1}{c^N - 1} c^{rank(c_i^t)-1}, i = 1, \ldots, N$$

donde $c = 0{,}3$.

Introducción a los algoritmos evolutivos
Cruce PMX

- Se realiza el cruce PMX con probabilidad p_c.

El cruce PMX consiste en, dados dos padres x_i^t y x_j^t, y dos puntos de cruce, por ejemplo:

$$x_i^t = (1,2,3|4,5,6,7|8,9)$$

$$x_j^t = (4,5,2|1,8,7,6|9,3)$$

se producen dos hijos $x_i'^t$ y $x_j'^t$, en principio ilegales, de la forma:

$$x_i'^t = (1,2,3|1,8,7,6|8,9)$$

$$x_j'^t = (4,5,2|4,5,6,7|9,3)$$

Introducción a los algoritmos evolutivos
Cruce PMX

Para reparar los hijos de forma que sean una permutación, se establecen los siguientes mapeos entre los elementos dentro de ambos segmentos de cruce:

$$1 \leftrightarrow 4, 8 \leftrightarrow 5, 7 \leftrightarrow 6, 6 \leftrightarrow 7$$

Se utilizan estos mapeos fuera de los segmentos de cruce, y los hijos resultantes son:

$$x_i'^t = (4, 2, 3|1, 8, 7, 6|5, 9)$$

$$x_j'^t = (1, 8, 2|4, 5, 6, 7|9, 3)$$

Introducción a los algoritmos evolutivos
Mutación por intercambio recíproco

- Se realiza la mutación por intercambio recíproco con probabilidad p_m.

La mutación por intercambio recíproco consiste en intercambiar dos genes entre sí, por ejemplo:

$$x_i^t = (4, \mathbf{2}, 3|1, 8, 7, 6|5, \mathbf{9})$$

$$\downarrow$$

$$x_i'^t = (4, \mathbf{9}, 3|1, 8, 7, 6|5, \mathbf{2})$$

Resultados con AE1

Resultados con AE1
Grid Search

Mejor resultado en el Grid Search con 3 repeticiones, $G = 100$ y $N = 30$:

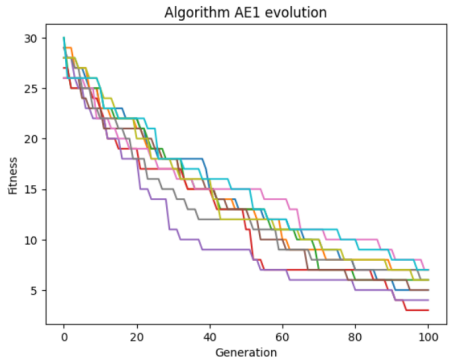
- Media del fitness = 4,3333
- $p_c = 0{,}45$
- $p_m = 0{,}2033$

Tiempo de ejecución: 16 minutos y 15 segundos.

Resultados con AE1
Nuevas ejecuciones con parámetros ajustados

Con 10 repeticiones, $G = 100$ y $N = 100$:

```
p_c: 0.45  p_m: 0.20333333333333334
Generation: 100 Fitness: 5.000000 : : 101it [00:10,  9.23it/s]
Generation: 100 Fitness: 7.000000 : : 101it [00:10, 10.00it/s]
Generation: 100 Fitness: 6.000000 : : 101it [00:09, 10.42it/s]
Generation: 100 Fitness: 3.000000 : : 101it [00:08, 11.54it/s]
Generation: 100 Fitness: 4.000000 : : 101it [00:16,  6.00it/s]
Generation: 100 Fitness: 5.000000 : : 101it [00:10,  9.84it/s]
Generation: 100 Fitness: 7.000000 : : 101it [00:11,  8.67it/s]
Generation: 100 Fitness: 6.000000 : : 101it [00:13,  7.64it/s]
Generation: 100 Fitness: 6.000000 : : 101it [00:11,  9.05it/s]
Generation: 100 Fitness: 7.000000 : : 101it [00:11,  8.93it/s]
Media:  5.6
Máximo:  7.0
Mínimo:  3.0
Desviación estándar:  1.2806248474865698
```



Tiempo de ejecución: 1 minuto y 55 segundos.

Resultados con AE2

Resultados con AE2
Grid Search

Mejor resultado en el Grid Search con 3 repeticiones, $G = 100$ y $N = 30$:

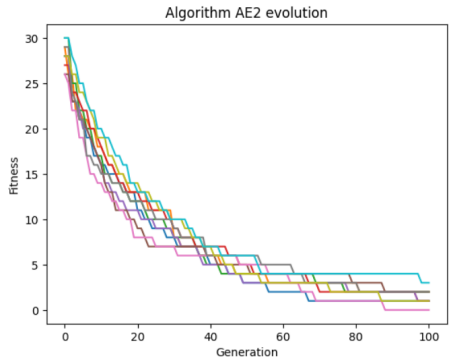
- Media del fitness = 4
- $p_c = 0{,}67$
- $p_m = 0{,}3$

Tiempo de ejecución: 15 minutos y 59 segundos.

Resultados con AE2
Nuevas ejecuciones con parámetros ajustados

Con 10 repeticiones, $G = 100$ y $N = 100$:

```
p_c: 0.67  p_m: 0.3
Generation: 100 Fitness: 1.000000 : : 101it [00:12,  8.24it/s]
Generation: 100 Fitness: 2.000000 : : 101it [00:11,  8.68it/s]
Generation: 100 Fitness: 2.000000 : : 101it [00:13,  7.75it/s]
Generation: 100 Fitness: 1.000000 : : 101it [00:10,  9.38it/s]
Generation: 100 Fitness: 1.000000 : : 101it [00:11,  8.90it/s]
Generation: 100 Fitness: 2.000000 : : 101it [00:11,  8.97it/s]
Generation: 88 Fitness: 0.000000 : : 89it [00:09,  9.77it/s]
Generation: 100 Fitness: 2.000000 : : 101it [00:11,  8.55it/s]
Generation: 100 Fitness: 1.000000 : : 101it [00:15,  6.64it/s]
Generation: 100 Fitness: 3.000000 : : 101it [00:19,  5.10it/s]
Media:  1.5
Máximo:  3.0
Mínimo:  0.0
Desviación estándar:  0.806225774829855
```



Tiempo de ejecución: 2 minutos y 7 segundos.

Tests estadísticos

Tests estadísticos
Mean

```
Mean
win: El algoritmo en la columna gana al algoritmo de la fila
loss: El algoritmo en la columna pierde frente al algoritmo de la fila
tie: El algoritmo en la columna empata con al algoritmo de la fila
                AE1       AE2
AE1               -       win
AE2            loss         -
Ranking              Wins     Losses     Wins-Losses
AE2                     1          0               1
AE1                     0          1              -1
```

- Gana AE2.

Tests estadísticos
Ranksum

```
Ranksum
win: El algoritmo en la columna gana al algoritmo de la fila
loss: El algoritmo en la columna pierde frente al algoritmo de la fila
tie: El algoritmo en la columna empata con al algoritmo de la fila
                AE1       AE2
AE1               -       win
AE2            loss         -
p-values
                           AE1           AE2
AE1                          -    0.00017929
AE2                0.00017929             -
Ranking              Wins     Losses     Wins-Losses
AE2                     1          0              1
AE1                     0          1             -1
```

- Gana AE2.
- p-value= $0{,}00017929 < 0{,}05 \Rightarrow$ diferencia estadísticamente significativa.

Tests estadísticos
Signed Rank

```
Signed Rank
win: El algoritmo en la columna gana al algoritmo de la fila
loss: El algoritmo en la columna pierde frente al algoritmo de la fila
tie: El algoritmo en la columna empata con al algoritmo de la fila
                AE1       AE2
AE1               -       win
AE2            loss         -
p-values
                          AE1           AE2
AE1                         -     0.00195312
AE2                 0.00195312             -
Ranking               Wins     Losses     Wins-Losses
AE2                      1          0               1
AE1                      0          1              -1
```

- Gana AE2.
- p-value= $0{,}00195312 < 0{,}05 \Rightarrow$ diferencia estadísticamente significativa.

Tests estadísticos
Total Mean, Total Ranksum y Total Signed Rank

Total Mean

Ranking	Wins	Losses	Wins-Losses
AE2	1	0	1
AE1	0	1	-1

Total Ranksum

Ranking	Wins	Losses	Wins-Losses
AE2	1	0	1
AE1	0	1	-1

Total Signed Rank

Ranking	Wins	Losses	Wins-Losses
AE2	1	0	1
AE1	0	1	-1

- Gana AE2.

Conclusiones

Conclusiones

AE2 ha mostrado un mejor desempeño que AE1 en todos los tests estadísticos, y además esta diferencia es significativa, lo que sugiere que usar selección por ranking exponencial (con $c = 0{,}3$) es una mejor opción que usar selección proporcional.

Además, la diferencia en tiempo de ejecución es mínima entre ambos algoritmos:

Algoritmo	Grid Search	Ejecuciones finales
AE1	16 minutos 15 segundos	1 minuto 55 segundos
AE2	15 minutos 59 segundos	2 minutos 7 segundos

Cuadro 1: Tiempos de ejecución de los algoritmos

Conclusiones

La selección proporcional asigna a cada individuo la siguiente probabilidad:

$$p_i^t = \frac{f(x_i^t)}{\sum_{i=1}^{N} f(x_i^t)}, i = 1, \ldots, N$$

- Si la población tiene individuos con valores de fitness muy desiguales, la selección proporcional puede centrarse exclusivamente en seleccionar un pequeño número de individuos altamente adecuados.
- Esto puede limitar la exploración, llevando a que el algoritmo quede atrapado en soluciones subóptimas.

Conclusiones

La selección por ranking exponencial asigna a cada individuo la probabilidad:

$$p_i^t = \frac{c-1}{c^N - 1} c^{rank(c_i^t)-1}, i = 1, \ldots, N$$

donde $c = 0{,}3$ (c cercano a 0 significa mayor número esperado de descendientes del mejor individuo).

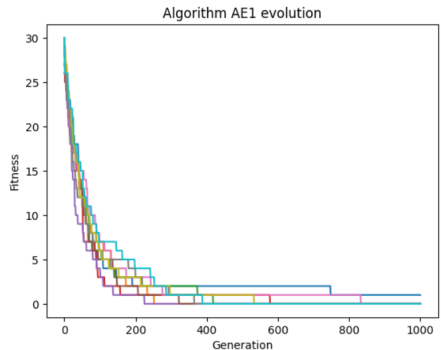
- Los individuos se ordenan en una lista de acuerdo a sus adecuaciones, y la probabilidad de selección de un individuo se obtiene en base a su posición en la lista ordenada.
- Incluso los peores individuos en términos de fitness tienen una oportunidad de ser seleccionados.
- Aunque se siguen priorizando los mejores individuos, el enfoque es más equilibrado que con la selección proporcional.

Pruebas extra

Pruebas extra
Ejecuciones de AE1 con 1000 *generaciones*

Con 10 repeticiones, $N = 100$ y aumentando el número de generaciones a $G = 1000$:

```
p_c: 0.45  p_m: 0.20333333333333334
Generation: 1000 Fitness: 1.000000 : : 1001it [01:35, 10.53it/s]
Generation: 252 Fitness: 0.000000 : : 253it [00:24, 10.12it/s]
Generation: 418 Fitness: 0.000000 : : 419it [00:44,  9.34it/s]
Generation: 578 Fitness: 0.000000 : : 579it [00:56, 10.28it/s]
Generation: 225 Fitness: 0.000000 : : 226it [00:23,  9.73it/s]
Generation: 322 Fitness: 0.000000 : : 323it [00:34,  9.43it/s]
Generation: 833 Fitness: 0.000000 : : 834it [01:11, 11.64it/s]
Generation: 365 Fitness: 0.000000 : : 366it [00:35, 10.39it/s]
Generation: 533 Fitness: 0.000000 : : 534it [00:54,  9.73it/s]
Generation: 388 Fitness: 0.000000 : : 389it [00:36, 10.63it/s]
Media:  0.1
Máximo:  1.0
Mínimo:  0.0
Desviación estándar:  0.30000000000000004
```

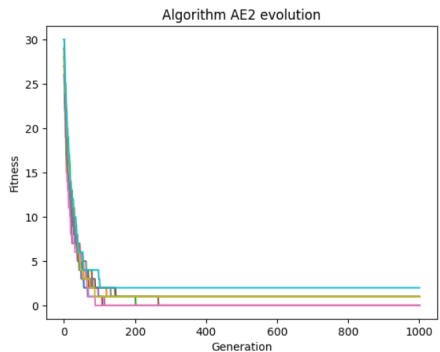


Tiempo de ejecución: 7 minutos y 58 segundos.

Pruebas extra
Ejecuciones de AE2 con 1000 *generaciones*

Con 10 repeticiones, $N = 100$ y aumentando el número de generaciones a $G = 1000$:

```
p_c: 0.67  p_m: 0.3
Generation: 1000 Fitness: 1.000000 : : 1001it [01:48,  9.18it/s]
Generation: 1000 Fitness: 1.000000 : : 1001it [01:39, 10.01it/s]
Generation: 202 Fitness: 0.000000 : : 203it [00:18, 10.69it/s]
Generation: 108 Fitness: 0.000000 : : 109it [00:10, 10.87it/s]
Generation: 114 Fitness: 0.000000 : : 115it [00:11, 10.10it/s]
Generation: 266 Fitness: 0.000000 : : 267it [00:25, 10.58it/s]
Generation: 88 Fitness: 0.000000 : : 89it [00:08, 11.11it/s]
Generation: 1000 Fitness: 1.000000 : : 1001it [01:42,  9.79it/s]
Generation: 1000 Fitness: 1.000000 : : 1001it [01:41,  9.87it/s]
Generation: 1000 Fitness: 2.000000 : : 1001it [01:34, 10.61it/s]
Media:  0.6
Máximo:  2.0
Mínimo:  0.0
Desviación estándar:  0.66332495807108
```



Tiempo de ejecución: 9 minutos y 42 segundos.

Pruebas extra
Conclusiones

- Con un número reducido de generaciones, AE2 funciona mejor ya que la selección por ranking exponencial favorece la exploración, lo que le permite disminuir el fitness de forma más efectiva en las primeras etapas.
- Cuando se aumenta el número de generaciones, una vez que se encuentra un buen individuo, con un fitness muy cercano a 0, AE1 acelera la propagación de este individuo dentro de la población. Así, la convergencia con AE1 es más rápida una vez que el óptimo es fácilmente alcanzable sin necesidad de demasiada exploración.
- En contraste, AE2 sigue favoreciendo la exploración incluso cuando ya se ha encontrado una buena solución, con un fitness muy cercano a 0. En lugar de centrarse en explotar ese individuo, sigue explorando otras soluciones, lo que puede retrasar la convergencia.


UNIVERSIDAD DE MURCIA
1272