Uso de algoritmos genéticos para búsqueda y evaluación de estrategias en HU-NLHE

TRABAJO DE FIN DE GRADO

Doble Grado en Ingeniería Informática y Matemáticas

Ignacio Funke Prieto

Dirigido por el Doctor

Manuel Núñez

Departamento de Sistemas Informáticos y Computación
Facultad de Informática
Universidad Complutense de Madrid

Junio 2015

Documento maquetado con TEXIS v.1.0+.

Uso de algoritmos genéticos para búsqueda y evaluación de estrategias en HU-NLHE

Trabajo de fin de grado

Ignacio Funke Prieto

Dirigido por el Doctor

Manuel Núñez

Departamento de Sistemas Informáticos y Computación
Facultad de Informática
Universidad Complutense de Madrid

Junio 2015

A Enrique y Arantxa

Agradecimientos

En primer lugar, quiero dar las gracias a mi director, Manuel Núñez, por haber conseguido despertar en mí el interés por el NLHE y por haberme guiado con soltura por el largo pero ameno camino que detallan estas páginas. Innumerables correos y reuniones, algunas de ellas a distancia, han ido dando forma a este Trabajo, donde sus aportaciones, mediante intrigantes ideas y consejos de inestimable valor ante las pequeñas piedras que iban apareciendo durante el camino, han sido piezas centrales en su composición.

Y quiero dar también las gracias a Marco Antonio y Pedro Pablo Gómez, profesores de esta facultad, por haber desarrollado y puesto a disposición del público TeXiS, una maravillosa plantilla de LaTeX para tesis doctorales y trabajos de fin de grado con la que he ahorrado mucho tiempo y esfuerzo durante la composición de este Trabajo en detalles que de otra forma habrían sido verdaderos quebraderos de cabeza.

Resumen

El NLHE es un juego muy atractivo, desde el punto de vista de la inteligencia artificial, pues se trata de un juego de información imperfecta, con una alta componente de azar a corto plazo y caracterizado por un espacio de estados de gran magnitud. Por ello, existen numerosos trabajos de investigación en busca de crear agentes inteligentes que sean capaces de jugar de forma óptima. En este trabajo presentamos una primera aproximación al mundo de la creación de agentes de poker y lo enfocamos a la aplicación de técnicas evolutivas, en concreto, al uso de algoritmos genéticos. Nos centraremos en un escenario más reducido al general, concretamente, en la variante en la que se restringe el número de jugadores a dos, Heads Up. Además, suponemos que las opciones a tomar en la mano se deciden preflop e imponemos limitaciones en la ronda de apuestas. En este marco, creamos jugadores con capacidades evolutivas que siguen estrategias sencillas y que evolucionamos hasta la solución buscada mediante un mecanismo de torneos. En este tipo de trabajo cobran especial importancia los mecanismos para estudiar gráficamente la exploración de los espacios de jugadores. Como resultado final, es posible concluir que los algoritmos genéticos se comportan bien a la hora de encontrar soluciones eficientes en este marco limitado, pero tienen dificultades a la hora de determinar decisiones óptimas conforme aumenta la complejidad de las decisiones que tienen que tomar los jugadores.

Palabras clave: Agentes de poker, Algoritmos genéticos, Heads Up, NLHE, Poker.

Abstract

In the field of Artificial Intelligence, NLHE is a very attractive game since it exhibits imperfect information, a significant element of chance and it features a large state space. Consequently, there exist a great deal of research work seeking the development of intelligent agents that play optimally. In this work, we give a humble approach to the world of poker agents, focusing on evolutionary techniques and specifically on genetic algorithms. We will restrict the original conditions and we will work with the two-player poker variant, Heads Up. In addition, all the strategical decisions will be made during the preflop stage and some limitations will be established in the betting round. In this context, we will define players with simple strategies and evolutionary abilities that will evolve by means of a tournament mechanism. In this work, they are particularly relevant all the mechanisms to study graphically the exploration of player spaces. As a result, it is possible to conclude that genetic algorithms are suitable to find efficient solutions in this limited framework, however, they find difficulties while trying to delimit optimal decisions as those decisions increase in complexity.

Keywords: Genetic algorithms, Heads Up, NLHE, Poker agents, Poker.

Índice

Índice de figuras

Índice de Tablas

Índice de algoritmos

Capítulo 1

Introducción

En este primer capítulo presentaremos nuestros objetivos de trabajo y trataremos de familiarizar al lector con los diferentes conceptos manejados a lo largo del mismo.

1.1. Introducción

Sin lugar a dudas, el poker es uno de los juegos de cartas que más renombre ha adquirido a lo largo de los últimos años. De entre todas sus variantes, la más popular y con más fuerza en la escena de la competición es el No-Limit Texas Hold'em o NLHE, cuya diferencia más notable con sus variantes hermanas estriba en que los jugadores apuestan esperando obtener la mejor combinación de cartas entre las de su mano y unas cartas comunes para todos en el centro de la mesa, conocidas como cartas de comunidad.

En líneas generales, el NLHE se sucede en una serie de fases, acompañadas por sendas rondas de apuestas, a lo largo de las cuales se van repartiendo las cartas hasta completarlas. En la primera de ellas, el *preflop*, se reparten las cartas de cada jugador; le sigue el *flop*, que abre con las tres primeras cartas de comunidad para posteriormente ser completadas con una carta en el *turn* y otra en el *river*. Durante las rondas de apuestas, los jugadores deciden si elevar la apuesta de la mesa, ver la apuesta o retirarse; finalizando cuando todos los jugadores han igualado la apuesta. Una vez completada la última fase, los jugadores enseñan sus cartas y el que obtenga mejores figuras se lleva el bote de apuestas.

Si nos enmarcamos en el mundo de la inteligencia artificial y la teoría de juegos, el NLHE resulta altamente atractivo, no sólo por su alta popularidad, sino por el reto que supone, dada su complejidad, su resolución. Se trata de un juego de información imperfecta con una alta componente de azar que, sumado al gigantesco espacio de estados que lo caracteriza, definen un juego en el que la investigación actual ha puesto celosamente su mira.

Los primeros pasos en su resolución giran en torno a una versión del mismo en la que sólo juegan dos jugadores, conocida como *Heads up*, y más concretamente, en la variante Limit del mismo, en el que la cantidad máxima que los jugadores pueden apostar está limitada. En este escenario, investigadores de la prestigiosa *Universidad de Alberta* han conseguido resolver débilmente el juego mediante

CFR$^+$[4], un algoritmo que hace una exploración ingeniosa del espacio de estados.

La investigación continúa para la versión No Limit con aproximaciones que tímidamente arañan la meta de resolverlo. En esta dirección, la *Universidad de Carnegie Mellon* ha conseguido desarrollar un robot, bautizado con el nombre de *Claudico*[1], que se ha permitido el lujo de desafiar a los mejores jugadores de poker del mundo, como ya lo hizo el ordenador *Deep Blue* de IBM contra Gary Kasparov en el terreno del ajedrez. Además, entre la comunidad de jugadores ya existen programas de ayuda a la toma de decisiones con resultados altamente competentes que permiten inclinar la balanza muy fácilmente del lado del que los emplea.

Pero todo esto no sería posible sin las enormes contribuciones del recientemente fallecido John Forbes Nash en el campo de la teoría de juegos, cuyas ideas son el esqueleto que vertebra la gran mayoría de las aproximaciones actuales y en las cuales humildemente se ha inspirado este presente trabajo.

1.2. Objetivos

En este trabajo presentaremos una primera aproximación al mundo de la creación de agentes de poker, centrándonos principalmente en la aplicación de técnicas evolutivas para obtener las mejores estrategias. Crearemos poblaciones de jugadores con estrategias parametrizables que evolucionarán hasta encontrar una estrategia óptima para las diferentes situaciones que plantearemos. Además, comprobaremos la efectividad de los algoritmos genéticos empleados comparando las soluciones obtenidas con soluciones de conocida efectividad para el escenario que usaremos.

Otro de los objetivos de este trabajo es buscar formas de transmitir información relativa a los jugadores y sus estrategias de forma muy visual durante todo el proceso de evolución. De esta manera podremos analizar de forma certera la exploración del espacio de jugadores realizada por los algoritmos genéticos.

Finalmente, desarrollaremos un marco de trabajo que permita una creación sencilla de agentes de poker y facilite las tareas de evolucionado y mostrado de información visual.

1.3. Conceptos

A modo de referencia, en esta sección se detallan algunos conceptos con los que el lector puede no estar habituado.

Ciega pequeña Apuesta inicial obligatoria que pone el jugador inmediatamente a la izquierda del crupier, que suele ser la mitad de la ciega grande. También usado para referirse a este mismo jugador.

Ciega grande Apuesta inicial obligatoria que pone el jugador inmediatamente a la izquierda de la ciega pequeña. También usado para referirse a este mismo jugador.

Cap Última subida posible en una ronda o límite superior que se puede apostar en una mano.

Raise Acción de elevar la apuesta de ese momento.

Call Acción de ver la apuesta de ese momento.

Preflop Primera etapa del NLHE. En ella se reparten las dos cartas de cada jugador.

Flop Segunda etapa del NLHE. En ella se reparten las tres primeras cartas de comunidad.

Turn Tercera etapa del NLHE. En ella se reparte la cuarta carta de comunidad.

River Última etapa del NLHE. En ella se reparte la quinta y última carta de comunidad.

1.4. El juego

El juego de poker en el que nos vamos a centrar es una versión reducida de la variante más popular en la actualidad: No-Limit Texas Hold'em. En concreto, consideraremos un escenario en el que sólo compiten dos jugadores, lo que se conoce como *Heads-Up* poker. Además, todo el juego se desarrollará en la primera etapa, en el *preflop*, antes de repartir las cartas de comunidad. Una vez repartidas las cartas de los dos jugadores, el jugador que empieza hablando, la *ciega pequeña*, tiene que decidir si apostar la cantidad máxima permitida (asumiremos un *cap* en las apuestas) o retirarse; el segundo jugador, la *ciega grande*, en caso de que el primer jugador haya apostado, tiene que decidir si ve la apuesta del primero o se retira.

Además, el stack efectivo, que es el mínimo entre las fichas disponibles de un jugador y la cantidad máxima apostable, será limitado a 10 ciegas o menos. En este aspecto, consideraremos dos escenarios distintos, uno en el que la cantidad máxima apostable es siempre fija en cada mano jugada, y otro en el que la cantidad máxima apostable se escoge aleatoriamente en cada mano jugada.

Hemos elegido esta modalidad tanto por su simplicidad como por tratarse de una situación que ya está muy analizada matemáticamente[3][7]. De esta forma podremos comprobar la efectividad de los algoritmos genéticos que emplearemos simplemente comparando con las soluciones ya calculadas para este escenario.

1.5. Consideraciones generales

A continuación se detallan algunos convenios y conceptos que utilizaremos ampliamente a lo largo de todo el trabajo.

	A	K	Q	J	T	9	8	7	6	5	4	3	2
A	20	20	20	20	20	20	20	20	20	20	20	20	20
K	20	20	20	20	20	20	20	20	20	20	20	19	19
Q	20	20	20	20	20	20	20	20	20	20	15	13	12
J	20	20	20	20	20	20	20	20	17	15	12	10	9
T	20	20	20	20	20	20	20	20	20	11	10	9	6
9	20	20	20	20	20	20	20	20	20	15	7	5	
8	20	16	13	12	17	20	20	20	20	20	11		
7	20	15	10	8	9	11	18	20	20	20	14		
6	20	14	9	6	5	5	8	14	20	20	20	10	
5	20	13	8	6						20	20	14	
4	20	12	7	5							20	11	
3	20	11	7	5								20	
2	20	11	7										14

Tabla 1.1: Ejemplo de tabla de manos

1.5.1. Las manos

La mano de un jugador son las dos cartas que recibe al comienzo de cada ronda. En lo sucesivo, una mano sólo se distinguirá de otra por los dos números de sus cartas y por si estas pertenecen al mismo palo o no. Los palos concretos no tendrán relevancia para las decisiones que tomen los jugadores.

Ello es así porque, a efectos prácticos, las figuras que se puedan obtener con cartas de distinto palo son independientes del palo de las mismas, ya que lo único que se tiene en cuenta en el cómputo final de figuras es si pertenecen al mismo palo o no. Es por ello que el caso de que sean del mismo palo se considera aparte, pues es más probable conseguir *color* si partimos de manos donde las dos cartas son del mismo palo.

1.5.1.1. Tablas de manos

En total hay 169 manos y es común representarlas en forma de tabla (véase Tabla 1.1). Las filas de la tabla representan el número de la primera carta en la mano ordenados por importancia (As, Rey, Reina,..., Dos); mientras que las columnas de la tabla representan el número de la segunda carta, también ordenados por importancia. Además, las celdas situadas por encima de la diagonal principal representan a manos con sus dos cartas del mismo palo, mientras que las celdas situadas por debajo de la diagonal principal representan a manos con las cartas de distinto palo.

Usaremos estas tablas para determinar la cantidad máxima apostable por un jugador en función de la mano que le haya tocado. Así pues, cada celda contendrá la cantidad máxima apostable, en ciegas, para la mano que representa.

	A	K	Q	J	T	9	8	7	6	5	4	3	2
A	∞	277	137	92	70	52	45	40	35	36	33	31	29
K	166	477	43	36	31	24	20	19	17	16	15	14	13
Q	96	29	239	25	22	16	13	11	11	10	10	9	8
J	68	25	16	160	18	13	10	9	7	7	7	6	6
T	53	23	15	12	120	11	9	7	6	5	5	4	4
9	41	18	12	9	7	96	8	6	5	4	3	3	3
8	36	15	10	7	6	5	80	6	5	4	3	2	2
7	31	14	9	6	5	4	4	67	4	3	3	2	2
6	28	13	8	5	4	4	3	3	48	3	2	2	2
5	28	12	8	5	4	3	2	2	2	49	2	2	2
4	26	11	7	5	3	2	2	2	2	2	41	2	1
3	24	11	6	4	3	2	2	2	1	1	1	33	1
2	23	10	6	3	2	2	1	1	1	1	1	1	24

Figura 1.1: Clasificación de Sklansky-Chubukov

1.5.1.2. Clasificaciones de manos

Una clasificación de manos es una ordenación de todas las posibles manos, según el convenio anteriormente citado, en función de una determinada característica. Existen en la literatura diversas clasificaciones, siendo la más popular la de Sklansky-Chubukov (véase Figura 1.1)[1], que puntúa cada carta en función de la cantidad máxima apostable con la que es rentable entrar con todo en el hipotético caso de que el oponente conozca nuestra mano. La clasificación que se utiliza de forma regular a lo largo de este trabajo es la *clasificación por fortaleza de mano*. En esta ordenación, una mano es más fuerte que otra, y por tanto es mejor que otra, si tiene una probabilidad mayor de ganar contra una mano aleatoria. Más concretamente, una mano es mejor que otra en esta ordenación, si enfrentada a muchas rondas en las que se han repartido aleatoriamente las cartas del contrario y las cartas de comunidad, obtiene mejores figuras que la segunda. Muchas de las estrategias que seguirán nuestros agentes de poker se basarán en esta clasificación y tomarán una decisión u otra en función de si la mano es relativamente buena o mala.

1.5.2. Solución de equilibrio

En teoría de juegos, una solución de equilibrio es una estrategia que es inexplotable, es decir, cualquier jugador que se salga de esa solución no va a conseguir obtener a largo plazo más beneficio del que ya consigue contra un oponente que se ciña también a la estrategia de equilibrio. En este escenario que hemos esco-

[1]Imagen original de `http://www.pokerlistings.es`

gido existe una solución de equilibrio. Es decir, existe un modo de operar, una estrategia conocida, que es inexplotable por cualquier otro jugador.

Uno de los objetivos en este trabajo será encontrar dicha solución de equilibrio, creando los jugadores y el entorno de evolución adecuados. Además, esta solución ya conocida servirá para contrastar muchos de los resultados obtenidos.

1.6. Acerca del marco de trabajo

Todos los algoritmos y experimentos que en los sucesivos capítulos se refieren han sido desarrollados en c++ con orientación a objetos. En este sentido, se ha desarrollado un marco de trabajo, mediante este lenguaje, orientado a facilitar la creación de agentes de poker con capacidades evolutivas y su posterior puesta a prueba mediante diferentes torneos. Además, se proveen herramientas para facilitar la creación de gráficas que muestren información visual relativa al proceso de evolución.

El marco hace un uso extenso de librerías externas como *poker-eval*[6], *pbots_ calc*[5] y *GALib*[8]; así como del software de graficado *Gnuplot*[2]. Las dos primeras se emplean para cálculo de probabilidades que involucren manos de poker, mientras que *GALib* provee de la estructura básica para la creación de un algoritmo genético. *Gnuplot*, a su vez, se ha utilizado para el mostrado de información visual. El código fuente está disponible para su consulta y utilización en `https://github.com/igofunke/NLHEEvol`.

1.7. Estructura del documento

El documento se ha estructurado en tres capítulos principales. El primero de ellos, *El algoritmo genético*, detalla cada una de las particularidades del algoritmo genético que hemos empleado. El segundo, *Definición de los jugadores*, nos guía a través del proceso de creación de los agentes de poker de este marco de trabajo. Y el tercero y último, *Experimentos*, recoge las pruebas a las que hemos sometido a los agentes de poker así como los resultados que hemos obtenido.

Capítulo 2

El algoritmo genético

En este capítulo describiremos con detalle el algoritmo genético que emplearemos para hacer evolucionar a nuestros jugadores. Cada una de las fases de un algoritmo genético común están particularizadas para adaptarse al problema de la evolución de jugadores parametrizables.

2.1. El material genético

Como hemos dicho anteriormente, nuestros jugadores seguirán estrategias parametrizables con un determinado número de parámetros. El material genético de cada tipo de jugador que definamos vendrá dado por estos parámetros y todas las operaciones de cruce y mutación sobre los jugadores operarán sobre dicho material genético. A modo de ejemplo, más adelante definiremos unos jugadores simples cuya estrategia se basará en dos parámetros reales r y c[1] comprendidos entre 0 y 1. El material genético de este jugador estará formado por esos dos genes.

2.2. Inicialización de la población

Al comienzo del algoritmo genético se crea una población inicial de tamaño previamente determinado. Para hacer una exploración efectiva del espacio de jugadores, necesitamos inicializar los jugadores con suficiente información como para que a lo largo de las diversas generaciones, mediante cruces y mutaciones, sean capaces de encontrar una solución óptima.

La inicialización dependerá del tipo de jugador escogido (véase Algoritmo 1), pues el material genético es distinto para cada jugador. Así pues, en el caso de los jugadores simples mencionados anteriormente, los dos parámetros r y c serán inicializados mediante una distribución uniforme $U(0, 1)$.

[1]El parámetro r denota la probabilidad de que la ciega pequeña haga *raise* mientras que c denota la probabilidad de que la ciega grande haga *call* ante una apuesta de la ciega pequeña.

Algoritmo 1 Inicialización de una población de jugadores

Input: Population $P = \{p_1, p_2, ..., p_n\}$
1: **procedure** INITIALIZE(P)
2: **for** $p \in P$ **do**
3: INITIALIZE(p)
4: **end for**
5: **end procedure**

2.3. Evaluación de la población: torneos

Una vez creada la población inicial, necesitamos una medida de lo efectivas que son las estrategias que cada uno de los individuos posee. Para ello, los jugadores de la población juegan un torneo y en función de cómo se desenvuelven en el torneo, se decide quiénes son los más aptos para evolucionar. Dependiendo del experimento que queramos realizar, tendremos varios tipos de torneos. En este trabajo, los torneos que usaremos son *todos contra todos* y *todos contra uno*.

En el torneo *todos contra todos*, enfrentamos a cada jugador de la población con el resto de jugadores. Se trata de una torneo de tipo *round-robin* en el que buscamos una medida del rendimiento global de un jugador con respecto a la población. Usaremos este tipo de torneo para encontrar soluciones de equilibrio para los distintos escenarios. Al evolucionar la población de forma conjunta, conseguimos que vayan aprendiendo estrategias que van siendo cada vez más inexplotables hasta converger a la solución de equilibrio. Cabe destacar la implementación que hemos escogido para un torneo round-robin (véase Algoritmo 2), la cual estaba basada en un sistema de planificación de calendarios de torneos conocido como *tablas de Berger*.

El otro tipo de torneo que usaremos es el torneo *todos contra uno* (véase Algoritmo 3). En este torneo enfrentaremos a toda la población de jugadores contra un único oponente. Lo que se busca con este tipo de torneo es encontrar a aquellos jugadores que usen estrategias más efectivas que la del oponente. De esta forma, por medio de generaciones sucesivas, logramos llegar a estrategias optimales.

En todos los casos, el funcionamiento de un torneo consiste, a su vez, en enfrentar a los jugadores por medio de partidas de poker simuladas. Dichas partidas se juegan hasta que uno de los jugadores se queda sin fichas o hasta que se agote un número de manos preestablecido.

Uno de los mayores problemas de este trabajo ha sido que el tiempo de ejecución de los torneos es demasiado grande. En el caso del torneo todos contra todos, el coste de ejecución es cuadrático en el número de jugadores y lineal en el número de manos simuladas. Por ello, todos los torneos se han paralelizado de la forma más eficientemente posible. Aún así, el coste de ejecución sigue siendo prohibitivo y una parte muy importante del esfuerzo dedicado a este trabajo se ha invertido en crear inicializadores, cruces y mutadores para los jugadores más eficientes, como veremos más adelante.

Algoritmo 2 Torneo round-robin para una población de jugadores

Input: Population $P = \{p_1, p_2, ..., p_n\}$

procedure TOURNAMENT(P)
  {Initializations}
  if EVEN(n) **then**
    $m \leftarrow n$, $start \leftarrow 0$
  else
    $m \leftarrow n + 1$, $start \leftarrow 1$
  end if
  {Round robin}
  for $round = 1..m - 1$ **do**
    for $i = start...(\frac{m}{2} - 1)$ **do**
      $i_1 \leftarrow i + (round - 1)\frac{m}{2}$ mód $(m - 1)$
      if $i = 0$ **then**
        $i_2 \leftarrow m - 1 - i$
      else
        $i_2 \leftarrow m - 1 - i + (round - 1)\frac{m}{2}$ mód $(m - 1)$
      end if
      MATCH(p_{i_1}, p_{i_2})
    end for
    Wait for matches to finish to continue to the next round
  end for
end procedure

Algoritmo 3 Torneo todos contra uno

Input: Population $P = \{p_1, p_2, ..., p_n\}$, opponent p^*

procedure TOURNAMENT(P, p^*)
  for $p \in P$ **do**
    MATCH(p, p^*)
  end for
end procedure

2.4. Función objetivo: El rendimiento de los jugadores

La función objetivo permite dar una puntuación a cada individuo de la población en función de lo interesantes que sean de cara a ser evolucionados. En nuestro caso, la función objetivo debe medir la eficiencia de la estrategia de un jugador determinado con respecto a los demás integrantes de la población. Para ello, hemos escogido el cociente entre el número de ciegas ganadas en total durante toda la ejecución del torneo y el número de manos jugados. De esta forma damos una métrica del rendimiento global del jugador para ese torneo en concreto.

Actualizamos las puntuaciones de los jugadores $c(p_1), c(p_2), ..., c(p_n)$ una vez que se termina el torneo, para poder trabajar con ellas en los siguientes pasos del algoritmo genético. Las puntuaciones pueden ser tanto negativas como positivas. Una puntuación negativa significará que el jugador ha perdido dinero tras acabar el torneo.

2.5. Función de fitness

Para trabajar con los distintos esquemas de selección de individuos, las puntuaciones asignadas a los mismos deben reunir ciertas características.

Para este algoritmo genético hemos usado el esquema de selección de la ruleta rusa, el cuál necesita trabajar con puntuaciones estrictamente mayores que 0. Para cumplir esta restricción, hemos aplicado una translación a las puntuaciones objetivo que deja la más negativa de ellas a 0. Si no hay ninguna puntuación objetivo negativa, estas se quedan igual (véase Algoritmo 4).

Algoritmo 4 Cálculo de las puntuaciones de fitness

Input: Population $P = \{p_1, p_2, ..., p_n\}$

```
1: procedure SCALE(P)
2:     m ← mín{0, c(p_1), c(p_2), ..., c(p_n)}
3:     for p ∈ P do
4:         f(p_i) ← c(p_i) + m
5:     end for
6: end procedure
```

2.6. Selección de los jugadores más fuertes

Una vez evaluado el rendimiento de los jugadores de la población, es el turno de seleccionar a los más fuertes para evolucionarlos. En este algoritmo utilizamos el esquema de selección de la ruleta rusa. En él, cada individuo tiene una probabilidad de ser seleccionado que es directamente proporcional a su puntuación de fitness.

La transformación lineal aplicada durante la fase de escalado respeta la ordenación de los jugadores trazada por las puntuaciones objetivo. De esta forma,

los jugadores que tienen más probabilidades de ser seleccionados son aquellos que mejor lo han hecho durante el torneo.

Durante esta fase se seleccionarán, por medio de este método, tantos individuos como tamaño tenga la población. Los individuos seleccionados serán cruzados y mutados para posteriormente pasar a formar parte de la nueva generación de jugadores.

2.7. Cruce de jugadores

Por cada dos individuos seleccionados, el algoritmo genético los cruza con cierta probabilidad. Los cruces permitirán intercambiar información de estrategia de un jugador a otro. Al cruzar dos jugadores entre sí buscamos que el material genético que los hace buenos en su estrategia se combine para obtener mejor material genético.

Cabe destacar que el cruce es específico para cada jugador y ejemplos de dichos cruces se verán más adelante en esta memoria. Sirvan, a modo de ejemplo, los jugadores simples parametrizados por los dos valores reales. En este caso, si se realiza el cruce, se intercambian el primer parámetro del primer jugador con el segundo parámetro del segundo jugador; manteniendo el resto de parámetros intactos.

2.8. Mutaciones

Una vez seleccionados los individuos se muta su material genético con cierta probabilidad. Las mutaciones permitirán hacer pequeños cambios a las estrategias para explorar otras posibilidades que pueden resultar más efectivas. Nuevamente, las mutaciones dependen del tipo de jugador escogido. En el caso de los jugadores simples, la mutación consiste en intercambiar los valores de los dos parámetros.

2.9. Terminación

Una vez cruzados y mutados los jugadores, obtenemos una nueva generación. El proceso anteriormente descrito se vuelve a repetir para dar lugar a las siguientes generaciones de jugadores que convergerán a la estrategia buscada.

Existen varios criterios de terminación, pero el que usamos ampliamente en este trabajo es el criterio de terminación por convergencia. Usando este criterio, el algoritmo genético convergerá cuando la función objetivo del mejor jugador de la población se mantenga estable durante un cierto número de generaciones. De esta forma, aseguramos que hemos llegado a una estrategia optimal, ya que esta es inmejorable.

2.10. Elitismo

En algunos experimentos usamos *elitismo*. Básicamente, el elitismo consiste en mantener los mejores jugadores de cada generación para la siguiente. De esta forma, nos aseguramos de que el material genético que ha resultado ser bueno en una generación no se pierda en la siguiente y podamos seguir trabajando sobre él.

Esta aproximación es particularmente interesante cuando trabajamos con jugadores con material genético muy grande. En este caso, el cruce tiene menos probabilidades de intercambiar bien el material genético y puede echar a perder una estrategia.

2.11. Esquema general del algoritmo genético

Finalmente, presentamos un esquema genérico del algoritmo genético que vamos a seguir (véase Algoritmo 5), a modo de esquema, para resumir las ideas y conceptos que utilizaremos en este trabajo. Algunas de las partes han sido descritas en las secciones anteriores mientras que profundizaremos más en otras, como la inicialización, el cruce y la mutación, en el capítulo siguiente.

Algoritmo 5 Esquema del algoritmo genético utilizado

```
1: procedure GeneticAlgorithm
2:     Create population P = {p_1, p_2, ..., p_n}
3:     Initialize(P)
4:     while {Termination criteria} do
5:         Tournament(P)
6:         {Player's performances {c(p_1], c(p_2), ..., c(p_n)} are updated as a consequence of the previous procedure}
7:         Scale(P)
8:         {Player's fitness scores {f(p_1), f(p_2), ..., f(p_n)} are updated as a consequence of the previous procedure}
9:         Select players using a Roulette Wheel Selection scheme
10:        Perform crossovers using player specific functions
11:        Perform mutations using player specific functions
12:    end while
13: end procedure
```

Capítulo 3

Definición de los jugadores

Como hemos descrito en el capítulo anterior, los individuos de nuestro algoritmo genético son jugadores que siguen estrategias parametrizadas. Cada vez que juegan un torneo, los jugadores se ciñen a sus estrategias y es su rendimiento global el que determina la efectividad de la estrategia. La estrategia de un jugador depende del estado del juego en cada momento. En nuestro escenario, la estrategia de los jugadores se va a basar únicamente en la posición que ocupa en la mesa, en las cartas de su mano y en la cantidad máxima apostable. Por último, para poder evolucionarlos, los jugadores tienen que tener definidos un inicializador, un cruce y una mutación. A continuación definiremos cada uno de los jugadores en estos términos.

3.1. Jugadores simples

Los jugadores simples basan su estrategia en apostar el máximo permitido cuando la mano que les ha tocado es *suficientemente buena*. Dicho de otra forma, si la mano está dentro de un porcentaje prefijado de las mejores en la clasificación por fortaleza, entonces realiza la apuesta.

Como ya hemos indicado, los jugadores que consideramos en este trabajo se encuentran en el escenario en el que la cantidad máxima apostable es fija y, por tanto, su estrategia sólo tendrá en cuenta la posición que ocupa (ciega pequeña o ciega grande) y las cartas de su mano.

3.1.1. Definición paramétrica

En este marco básico los jugadores están definidos por dos parámetros, r y c, que como ya hemos indicado determinan la probabilidad con la que apuestan desde la ciega pequeña y aceptan la apuesta desde la ciega grande, respectivamente. De ese modo, al tomar la decisión, si la mano que les ha tocado está dentro de las $r\,\%$ o $c\,\%$ mejores en la clasificación por fortaleza, entran en el bote; de lo contrario, se retiran.

3.1.2. Inicializador

Para inicializar los jugadores simples daremos un valor aleatorio a los dos parámetros reales r y c mediante una distribución uniforme $U(0,1)$ (véase Algoritmo 6).

Algoritmo 6 Inicializador de los jugadores simples

Input: Player $p = (r, c)$
1: **procedure** INITIALIZE(p)
2: $r \leftarrow U(0,1)$
3: $c \leftarrow U(0,1)$
4: **end procedure**

Con una población suficientemente grande, con este inicializador los jugadores tienen suficiente información entre todos como para, mediante cruces y mutaciones, explorar todo el espacio de jugadores simples, el cuál cubre todas las combinaciones reales de r y c.

3.1.3. Cruce

Para cruzar dos jugadores simples es suficiente intercambiar el parámetro r del primero de ellos con el parámetro c del segundo de ellos, dejando el otro parámetro intacto. Este proceso da lugar a dos hijos (véase Algoritmo 7).

Algoritmo 7 Cruce de los jugadores simples

Input: Player father $f = (r^f, c^f)$, player mother $m = (r^m, c^m)$
1: **function** CROSSOVER(f, m)
2: $(r^1, c^1) \leftarrow (r^f, c^m)$
3: $(r^2, c^2) \leftarrow (r^m, c^f)$
4: **return** $(r^1, c^1), (r^2, c^2)$
5: **end function**

Si dos jugadores tienen estrategias *buenas*, el cruce permite mezclar su material genético para encontrar, posiblemente, una estrategia mejor.

3.1.4. Mutador

La mutación de los jugadores simples consiste en intercambiar sus dos parámetros r y c (véase Algoritmo 8). La mutación, junto con el cruce, permite explorar todo el espacio de jugadores simples por medio de intercambios sucesivos de parámetros.

3.1.5. Gráficas

Para poder mostrar información visual sobre la evolución de poblaciones de jugadores simples usamos una nube de puntos. En la nube de puntos, cada individuo está representado por un punto. El eje de abscisas representa el valor del

Algoritmo 8 Mutación de los jugadores simples

Input: Player $p = (r, c)$
1: **procedure** MUTATION(p)
2: $(r, c) \leftarrow (c, r)$
3: **end procedure**

parámetro r mientras que el eje de ordenadas representa el valor del parámetro c (Figura 3.1).

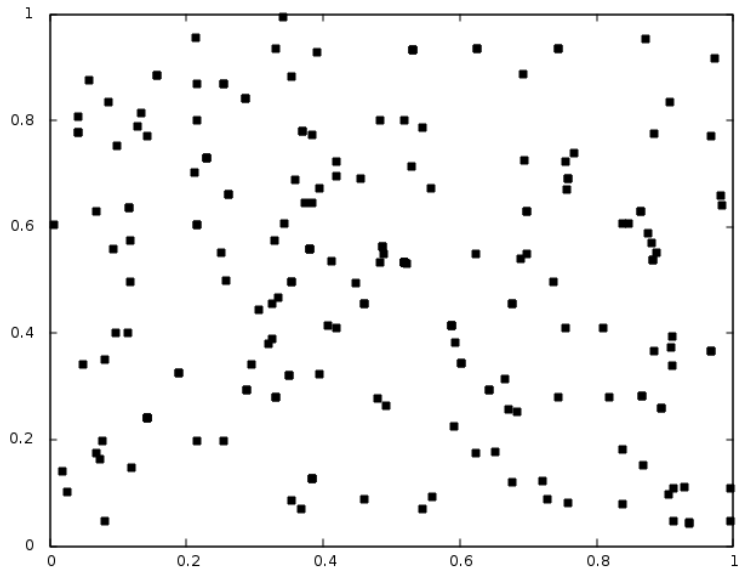


Figura 3.1: Nube de puntos para una población de jugadores simples

En cada generación del algoritmo, la nube de puntos se actualiza con los nuevos individuos. Este proceso permite ver, de un rápido vistazo qué individuos son fácilmente descartados y hacia dónde está convergiendo la población. En el Apéndice A mostramos una evolución de una población de jugadores simples. Podemos apreciar como la población se va concentrando alrededor de la solución obtenida.

3.2. Jugadores complejos en un escenario de stack efectivo fijo

Los jugadores complejos son una extensión de los jugadores simples. En vez de guiarse por una clasificación de manos, estos jugadores se basan en una tabla de manos. Para cada mano consideran si el movimiento consistente en entrar en el bote es EV+[1].

[1]Del término inglés *Expected Value*, esta expresión se utiliza habitualmente en el mundo del poker para denotar acciones que, matemáticamente y teniendo en cuenta la probabilidad de las distintas alternativas, deberían incrementar el número de fichas del jugador que las realiza.

Obviamente, los jugadores simples son un subconjunto de los jugadores complejos, pues si marcamos las k mejores manos de la clasificación por fortaleza como manos en las que se apuesta y marcamos el resto como manos en las que no merece la pena apostar, obtenemos la misma estrategia que los jugadores simples. Lo que se pretende con estos jugadores es eliminar las restricciones que supone basarse en las k mejores manos de una clasificación y añadir flexibilidad a cómo gestionar cada mano.

Veremos más adelante que la estrategia basada en la clasificación por fortaleza, la de los jugadores simples, es bastante efectiva, dado que mostraremos que los algoritmos genéticos devuelven unos jugadores complejos con estrategias muy parecidas a la de los jugadores simples, pese a la eliminación de la restricción.

Finalmente, es necesario comentar que mantendremos el escenario en el que la cantidad máxima apostable es fija y, por tanto, en la estrategia de estos jugadores no entra en juego dicho parámetro.

3.2.1. Definición paramétrica

Cada jugador complejo está equipado con dos tablas de manos $(r_{ij})_{1\leq i,j\leq 13}$ y $(c_{ij})_{1\leq i,j\leq 13}$ que toman valores en $\{0,1\}$ y determinan si se apuesta y si se acepta una apuesta, desde la ciega pequeña y desde la ciega grande, respectivamente, con la mano (i,j).

3.2.2. Inicializador

Para inicializar estos jugadores, primero se pensó en dar un valor aleatorio uniforme $U(0,1) \in \{0,1\}$ a cada mano de cada tabla. Los jugadores partían con información suficiente para que, mediante cruces y mutaciones, pudieran explorar todo el espacio de jugadores complejos. Sin embargo, los jugadores no partían con mucha inteligencia y, en seguida, convergían a óptimos locales con estrategias claramente poco eficaces. El hecho de que los algoritmos no se hayan podido correr con un número elevado de jugadores, debido al coste algorítmico prohibitivo, hacía que la diversidad de estrategias no fuera lo suficientemente grande como para tener individuos con estrategias medianamente diferenciadas que marcaran la diferencia. Como consecuencia de ello, la población se estancaba en estrategias poco útiles.

Teniendo en cuenta esta problemática se decidió dotar a los jugadores de cierta inteligencia inicial. El inicializador elegido está inspirado en el de los jugadores simples, pues inicializa las $r\,\%$ y $c\,\%$ mejores manos en (r_{ij}) y (c_{ij}) a 1 y el resto a 0, con r y c obtenidos aleatoriamente según una uniforme $U(0,1)$ (véase Algoritmo 9).

De este modo, los jugadores parten con mejores estrategias y son los cruces y mutadores los que se encargan de alterar estas estrategias de partida.

3.2.3. Cruce

El cruce de dos jugadores complejos consistirá en cruzar la tablas r y c de cada jugador entre sí. Dado que el proceso es idéntico para las dos tablas, en esta sección explicaremos únicamente el cruce de una de ellas.

Algoritmo 9 Inicializador de los jugadores complejos

Input: Player $p = (r_{ij}, c_{ij})$

```
1: procedure INITIALIZE(p)
2:     r ← U(0,1)
3:     c ← U(0,1)
4:     for (i,j) ∈ Hands do
5:         r_ij, c_ij ← 0
6:         if (i,j) ∈ BESTHANDS(r) then
7:             r_ij ← 1
8:         end if
9:         if (i,j) ∈ BESTHANDS(c) then
10:            c_ij ← 1
11:        end if
12:    end for
13: end procedure
```

En un principio, se consideró utilizar el cruce habitual para matrices. Este cruce consiste en seleccionar una celda al azar y mezclar las primeras celdas hasta esa celda del primer jugador con las celdas desde esa hasta la última del segundo jugador. Las celdas restantes se combinan de la misma forma para formar el segundo hijo. El problema de este cruce es que, al intercambiar tantos parámetros contiguos, puede echar a perder fácilmente una estrategia. El buen material genético puede estar localizado en zonas más puntuales de la tabla y por ello necesitábamos un cruce que intercambiara información de forma más local y puntual.

El cruce final recorre todas las celdas de la tabla de un jugador y, con cierta probabilidad, las intercambia con la celda homóloga del segundo jugador (véase Algoritmo 10). Con esto conseguimos un intercambio de información menos agresivo y más puntual.

3.2.4. Mutación

La mutación en un jugador complejo recorre todas las celdas de una tabla y, con cierta probabilidad, las intercambia con otra celda elegida al azar de la misma tabla (véase Algoritmo 11). De nuevo, conviene remarcar que el objetivo de la mutación consiste en introducir ligeras variaciones en la estrategia del jugador, de tal forma que se exploren soluciones que puedan resultar mejores.

3.2.5. Gráficas

En este nuevo marco de trabajo no podemos recurrir al plano cartesiano para mostrar información visual sobre los jugadores complejos. Ello es así dado que necesitamos representar 169 parámetros distintos. La alternativa que surgió fue utilizar *mapas de calor*. Esta representación gráfica es idónea para mostrar la información de tabla, pues están distribuidos de la misma forma y permiten ver, de un rápido vistazo, el valor de cada una de las celdas.

Algoritmo 10 Cruce de los jugadores complejos

Input: Player father $f = (r^f_{ij}, c^f_{ij})$, player mother $m = (r^m_{ij}, c^m_{ij})$

function CROSSOVER(f, m)
  for $(i, j) \in Hands$ **do**
    if FLIPCOIN($pCross$) **then**
      $r^1_{ij} \leftarrow r^m_{ij}$, $r^2_{ij} \leftarrow r^f_{ij}$
    else
      $r^1_{ij} \leftarrow r^f_{ij}$, $r^2_{ij} \leftarrow r^m_{ij}$
    end if
    if FLIPCOIN($pCross$) **then**
      $c^1_{ij} \leftarrow c^m_{ij}$, $c^2_{ij} \leftarrow c^f_{ij}$
    else
      $c^1_{ij} \leftarrow c^f_{ij}$, $c^2_{ij} \leftarrow c^m_{ij}$
    end if
  end for
  return $(r^1_{ij}, c^1_{ij}), (r^2_{ij}, c^2_{ij})$
end function

Algoritmo 11 Mutador de los jugadores complejos

Input: Player $p = (r_{ij}, c_{ij})$

procedure MUTATION(p)
  for $(i, j) \in Hands$ **do**
    if FLIPCOIN($pMut$) **then**
      $(i', j') \leftarrow$ RANDOMHAND()
      SWAP($r_{ij}, r_{i'j'}$)
    end if
    if FLIPCOIN($pMut$) **then**
      $(i', j') \leftarrow$ RANDOMHAND()
      SWAP($c_{ij}, c_{i'j'}$)
    end if
  end for
end procedure

En el caso de los jugadores que estamos considerando usamos dos tipos de gráficos basados en mapas de calor para mostrar la información relativa a la evolución de los jugadores a lo largo del algoritmo genético. El primer gráfico muestra los mapas de calor asociados a las tablas r y c del mejor jugador de la población (Figura 3.2). El color blanco indica que el jugador va a entrar en el bote con esa mano mientras que el color negro indica que no lo va a hacer. De esta forma, podemos ver cómo está evolucionando la estrategia más fuerte en cada generación.

Dado que esta representación no permite ver cómo están evolucionando el resto de jugadores de la población, el segundo gráfico es un mapa de calor compuesto por todos los mapas de calor de todos los jugadores de la población (Figura 3.3). Este gráfico es muy útil para observar la repetición de patrones entre diferentes miembros de la población.

La contrapartida de este último gráfico es que no se permite ver con tanto detalle la evolución de cada uno de los jugadores como con el primer gráfico. Por este motivo usaremos una combinación de los dos gráficos que hemos mencionado para mostrar la evolución de una población. En el Apéndice B mostramos una evolución de jugadores complejos.

3.3. Jugadores complejos en un escenario de stack efectivo variable

Estos jugadores son, a su vez, una extensión de los jugadores anteriores. Su estrategia se basa también en tablas de manos, pero distinguen, además, la cantidad máxima que pueden apostar para cada mano. Este tipo de jugadores está pensado para un escenario más realista en el que la cantidad máxima apostable varía con cada mano.

3.3.1. Definición paramétrica

En este caso, en las tablas de manos llevamos también la información de la cantidad máxima (hasta un máximo de 10 ciegas) que el jugador estaría dispuesto a apostar con la mano que recibe. Así pues, estos jugadores están parametrizados por dos tablas $(r_{ij})_{1<=i,j<=13}$ y $(c_{ij})_{1<=i,j<=13}$ que toman valores en $\{1, 2, ..., 10\}$ ciegas. Como en el caso anterior, la tabla r se usa para tomar las decisiones desde la ciega pequeña y la tabla c para las decisiones desde la ciega grande.

3.3.2. Inicializador

El primer inicializador que se utilizó en esta variante se servía de la misma filosofía que el primer inicializador de los jugadores anteriores. Consistía, por tanto, en inicializar todos los valores de las tablas a un valor aleatorio según una uniforme $U(1, 10)$. No obstante, como en el caso anterior, al no poder emplear poblaciones tan elevadas en número, las soluciones se estancaban en óptimos locales con estrategias poco definidas donde la poca diversidad de la población no ayudaba a explorar de forma eficaz el espacio de jugadores.

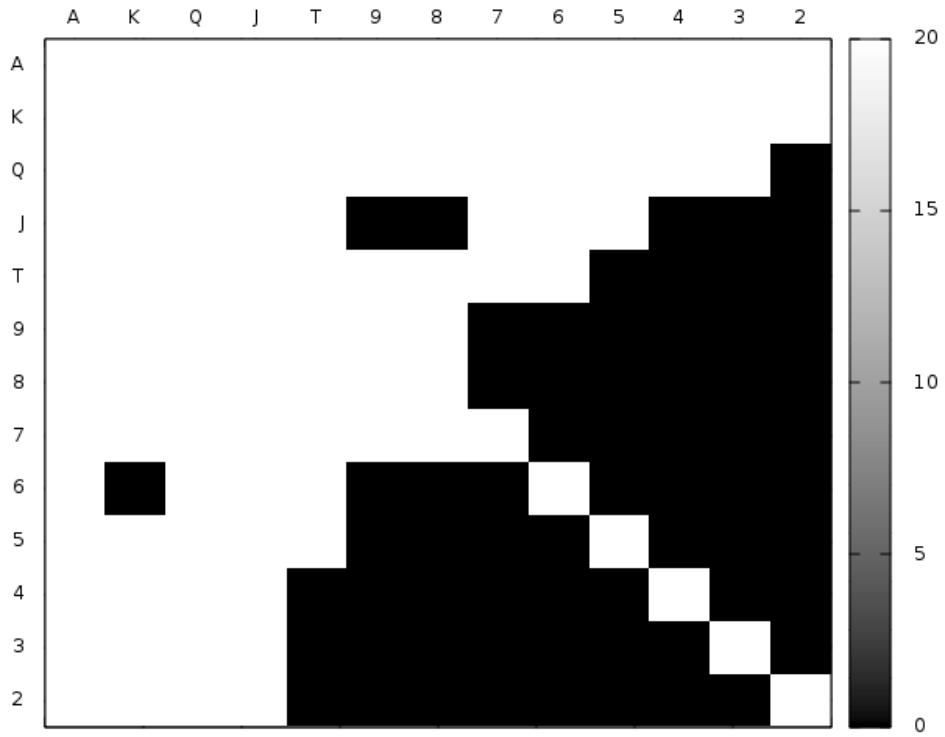


Figura 3.2: Mapa de calor para la tabla de manos en la posición de la ciega grande del mejor jugador de la población

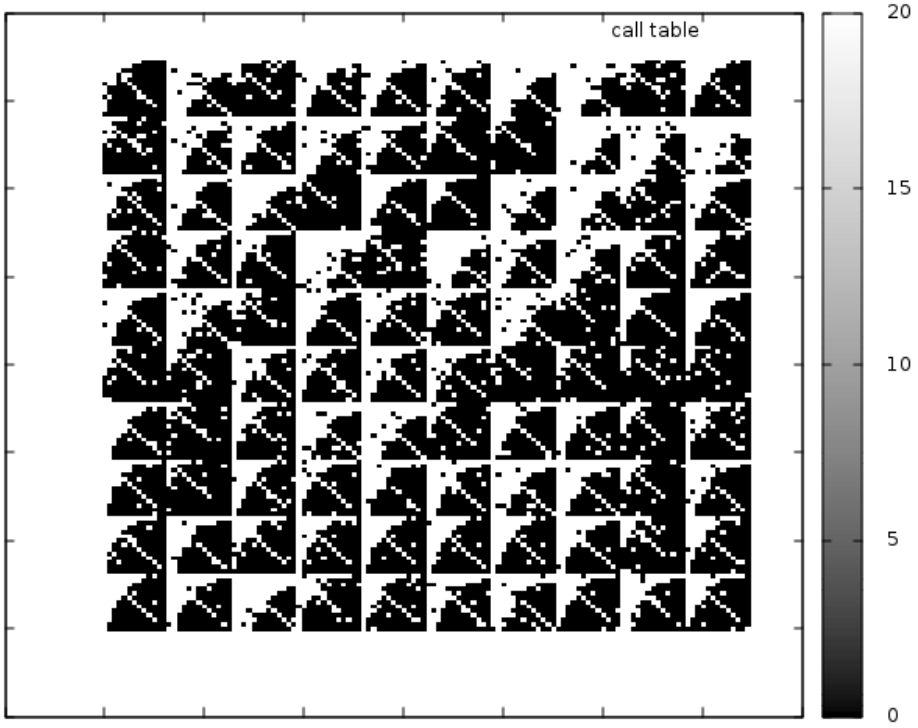


Figura 3.3: Mapa de calor de la población global. Tablas de manos en la posición de la ciega pequeña

Teniendo en cuenta estas limitaciones, el siguiente intento fue utilizar un inicializador también inspirado en el segundo inicializador de los anteriores jugadores. De esta forma, se inicializaban las $r\,\%$ mejores manos en la tabla r a un valor aleatorio dado por una uniforme $U(1, 10)$ igual para todas. Una situación similar ocurría para la tabla c. En este caso, los jugadores cuyas manos estaban inicializadas con valores bajos quedaban rápidamente descartados, pues jugaban muchas menos manos que los jugadores con valores altos y la población perdía diversidad muy rápidamente.

Finalmente, hemos optado por utilizar exactamente el mismo inicializador que los anteriores. Esto es, inicializa unas $r\,\%$ y $c\,\%$ mejores manos a 10 ciegas, el límite del máximo apostable, y el resto a 0 (véase Algoritmo 12). Los jugadores parten con la inteligencia necesaria para seguir estrategias lo suficientemente buenas y confían la exploración del resto del espacio a la aplicación de cruces y mutaciones.

Algoritmo 12 Inicializador de los jugadores complejos con cantidad máxima apostable variable

Input: Player $p = (r_{ij}, c_{ij})$

```
1: procedure INITIALIZE(p)
2:     r ← U(0,1)
3:     c ← U(0,1)
4:     for (i,j) ∈ Hands do
5:         r_ij, c_ij ← 0
6:         if (i,j) ∈ BESTHANDS(r) then
7:             r_ij ← 10
8:         end if
9:         if (i,j) ∈ BESTHANDS(c) then
10:            c_ij ← 10
11:        end if
12:    end for
13: end procedure
```

3.3.3. Cruce

Para el cruce de estos jugadores usamos exactamente el mismo cruce que para los anteriores. Dado que el material genético prometedor puede seguir localizado de forma puntual en diferentes zonas de las tablas, este cruce sigue siendo el más acertado para hacer un intercambio de información entre jugadores. La única diferencia es que, al intercambiar las celdas, ahora se intercambian cantidades máximas apostables y no simplemente el hecho de apostar o no.

El cruce, en este caso, no permite explorar todo el espacio de jugadores si partimos de los inicializadores descritos anteriormente. Con el inicializador, los jugadores sólo parten con información de con qué manos apostar, pero no de con qué cantidad máxima apostable hacerlo. De esta forma, si sólo confiáramos en el cruce, no podríamos explorar otras cantidades máximas apostables y, por ello, delegamos esta exploración en la mutación.

3.3.4. Mutador

Con el inicializador escogido, el mutador juega ahora un papel muy importante en el sentido de que debe explorar estrategias que varíen en la cantidad máxima apostable para cada mano. La primera idea que surgió fue la de mutar las celdas con cierta probabilidad asignándoles un valor aleatorio según una uniforme $U(1, 10)$.

El problema con este mutador es que hacía cambios muy agresivos en la estrategia de un jugador. Una celda podía ver disminuida o aumentada su cantidad máxima apostable de forma significativa y, como ha quedado mostrado posteriormente a la vista de los experimentos, estos jugadores son muy sensibles a cambios fuertes en su estrategia, lo que hacía que fácilmente fueran eliminados de la población.

La solución estaba, pues, en hacer cambios menos agresivos en la estrategia del jugador. El mutador final muta cada celda con cierta probabilidad, disminuyendo o aumentando su valor en una o dos unidades a lo sumo (véase Algoritmo 13).

Algoritmo 13 Mutador de los jugadores complejos con cantidad máxima apostable variable

Input: Player $p = (r_{ij}, c_{ij})$

```
1: procedure MUTATION(p)
2:     for (i, j) ∈ Hands do
3:         if FLIPCOIN(pMut) then
4:             r_ij ← r_ij + U(−2, 2)
5:         end if
6:         if FLIPCOIN(pMut) then
7:             c_ij ← c_ij + U(−2, 2)
8:         end if
9:     end for
10: end procedure
```

De esta forma, las estrategias derivadas se diferenciaban mínimamente de las anteriores pero lo suficiente como para determinar si ese camino de exploración es adecuado o no y construir sobre él.

3.3.5. Gráficas

Para estos jugadores seguimos teniendo los mapas de calor para el mejor jugador de la población (Figura 3.4) y los mapas de calor para la población global (Figura 3.5). La diferencia con los anteriores mapas es que ahora la cantidad máxima apostable para cada mano está representada en cada celda mediante una escala de grises. De esta forma, cuanto más cerca esté una celda del blanco, más fichas se permite apostar el jugador para esa mano concreta. Al igual que hemos hecho en los casos anteriores, en el Apéndice C mostramos un ejemplo de evolución de estos jugadores, en el cual se aprecia el papel que juegan estas gráficas.

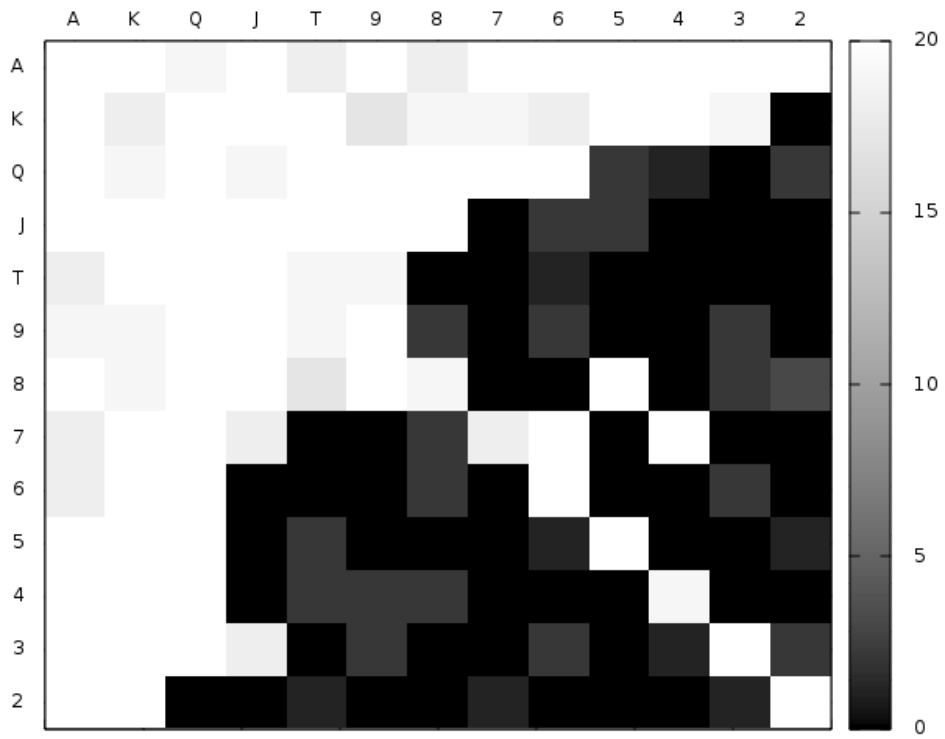


Figura 3.4: Mapa de calor para la tabla de manos en la posición de la ciega pequeña del mejor jugador de la población

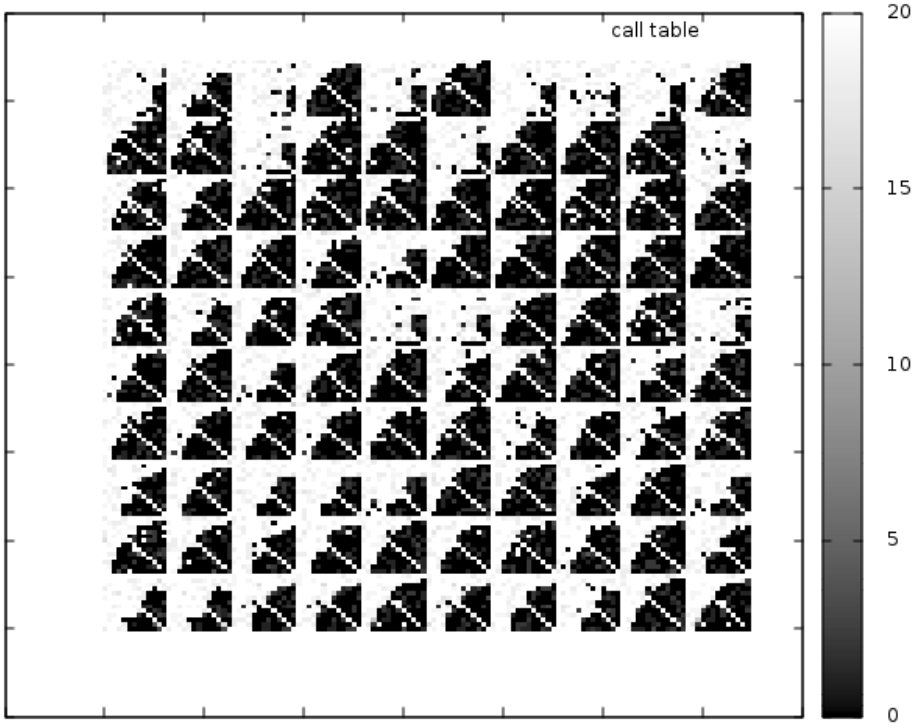


Figura 3.5: Mapa de calor de la población global. Tablas de manos en la posición de la ciega grande

Capítulo 4

Experimentos

En este capítulo presentamos, y analizamos los resultados obtenidos, los experimentos más relevantes que hemos realizado en el marco de este Trabajo de Fin de Grado. Estos son *uno contra uno*, *solución de equilibrio para jugares simples*, *modo adaptativo para jugadores simples*, *solución de equilibrio para jugadores complejos*, *modo adaptativo para jugadores complejos* y *solución de equilibrio para jugadores complejos en un escenario de stack efectivo variable*.

4.1. Uno contra uno

Este experimento enfrenta a dos jugadores a una partida de poker según el escenario que estamos usando en este trabajo. Está diseñado principalmente para comprobar el rendimiento de las soluciones obtenidas en los siguientes experimentos. Para configurarlo, se escogen dos jugadores de cualquiera de los tipos descritos en el capítulo anterior y se parametrizan según se desee. Acto seguido, se configura el número de manos a jugar y se ejecuta el experimento.

Durante la ejecución del experimento se llevan a cabo un número prefijado de partidas de poker (véase Figura 4.1). En cada repetición se muestra el número de manos jugadas (Línea *1), el rendimiento en bb/hands del primer jugador respecto al segundo (Línea *2), el porcentaje de veces que hacen *raise* (Línea *3), esto es, que entran al bote con todo desde la ciega pequeña; y el porcentaje de veces que hacen *call* (Línea *4), es decir, que ven la apuesta del primero desde la ciega grande. De esta forma, se puede comprobar la agresividad de cada jugador y su rendimiento relativo al otro. Además, al terminar, se muestra la media de los rendimientos (Línea *5).

Con este experimento podremos comprobar cómo la solución de equilibrio de los experimentos siguientes es realmente una solución de equilibrio o cómo las soluciones obtenidas para los experimentos en los que se busca una estrategia óptima frente a otra dada son realmente mejores.

Por último, es posible asignar una gráfica de rendimiento a este experimento. La gráfica se asocia a uno de los jugadores, permitiendo mostrar cómo se desenvuelve a lo largo de las manos que juega (véase Figura 4.2). En el eje horizontal representamos el número de manos jugadas mientras que en el eje vertical representamos el rendimiento hasta ese momento en bb/hands.

```
Match 0
-----------------------------------------------------
Hands played: 33853                                  (*1)
Player 1 performance: 1.48 bb/hands                  (*2)
Player 1 raised: 59.16%, Player 2 raised: 71.31%     (*3)
Player 1 called: 36.51%, Player 2 called: 36.32%     (*4)
-----------------------------------------------------
Match 1
-----------------------------------------------------
Hands played: 5749
Player 1 performance: 8.68 bb/hands
Player 1 raised: 59.36%, Player 2 raised: 72.17%
Player 1 called: 37.64%, Player 2 called: 35.70%
-----------------------------------------------------

[...]

Match 9
-----------------------------------------------------
Hands played: 3759
Player 1 performance: 13.30 bb/hands
Player 1 raised: 62.11%, Player 2 raised: 73.30%
Player 1 called: 35.20%, Player 2 called: 35.82%
-----------------------------------------------------
Player 1 performance mean: 6.019550 bb/hands         (*5)
```

Figura 4.1: Salida del experimento uno contra uno

4.1.1. El problema de la varianza

Si ejecutamos una partida entre dos jugadores hasta que estos se queden sin fichas, su rendimiento final varía enormemente con cada simulación. Como resultado de ello, las puntuaciones que obtenían los jugadores en algunos de los experimentos siguientes no eran en absoluto fiables y conducían a soluciones erróneas.

Si eliminamos la restricción de terminar la partida cuando se quedan sin fichas, haciendo que jueguen un número exacto de manos, podemos observar cómo disminuye la varianza del rendimiento de los jugadores conforme aumentamos el número de manos, como debe suceder. Sin embargo, para hacer más real el escenario, esa limitación debe estar ahí, lo que significa que no podemos hacer la varianza de los datos tan pequeña como queramos.

La forma de solucionar este problema consiste en simular varias partidas y calcular la media de los rendimientos. En este experimento se provee una media de los rendimientos al final de su ejecución (véase Figura 4.2) y podemos comprobar cómo la media de los rendimientos, para un número suficiente de simulaciones, es efectivamente suficientemente poco variable como para considerarse una métrica fiable en los torneos del algoritmo genético.

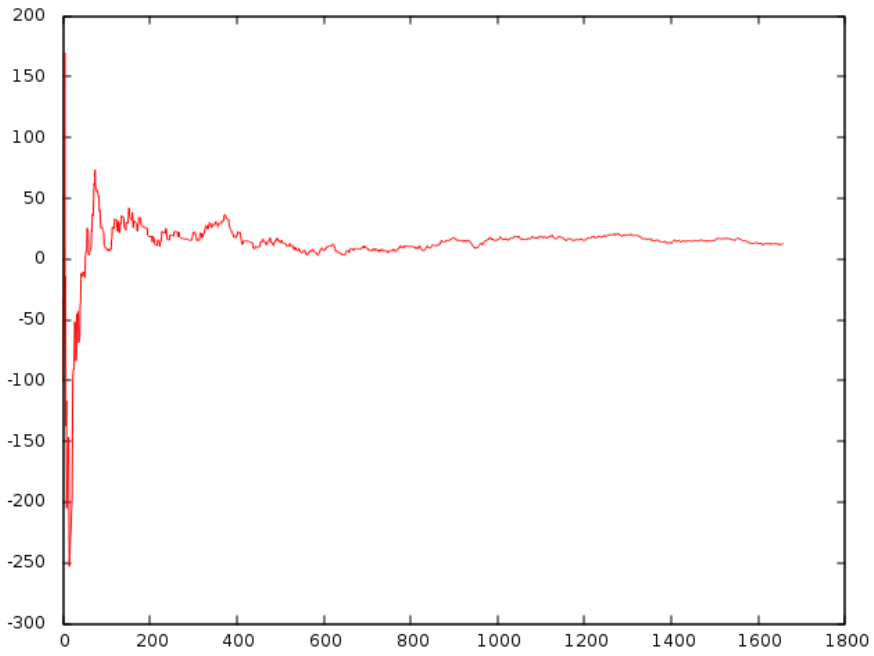


Figura 4.2: Gráfico de rendimiento de jugador en experimento uno contra uno

No obstante, en algunos experimentos, como los que involucran torneos todos contra todos, esto no es necesario. En esta situación la varianza del rendimiento se cancela con las sucesivas partidas que va jugando un jugador con el resto de jugadores. Esta peculiaridad es especialmente importante para no aumentar el coste de ejecución del algoritmo significativamente.

4.2. Solución de equilibrio con jugadores simples

Nuestro objetivo con este experimento es encontrar la solución de equilibrio para los jugadores simples. Usando el programa ICMizer podemos ver que la solución de equilibrio cuando restringimos la apuesta a un total de diez ciegas consiste en que la ciega pequeña apuesta el 57 % de sus mejores manos mientras que la ciega grande acepta la apuesta con el 37 % de las mejores manos (véase Figura 4.3). Dado que utilizamos una clasificación de manos algo diferente a la usada en ICMizer y que las poblaciones, mediante la evolución inducida, pueden alcanzar un resultado *bastante bueno* pero no alcanzar el óptimo, en nuestros experimentos no siempre obtenemos este resultado, pero la diferencia es despreciable.

Enfrentaremos a una población de jugadores en un torneo de tipo round robin e iremos evolucionando a los más fuertes hasta encontrar la solución buscada. Al enfrentar a los jugadores todos contra todos, aquel jugador que juegue en equilibrio es el que obtendrá un mejor rendimiento global relativo al resto de la población. De esta forma, los jugadores de equilibrio son los que sobrevivirán a lo largo de las generaciones y son las soluciones que obtendremos al finalizar el experimento.

El inicializador, el cruce y la mutación son los definidos anteriormente para este tipo de jugadores. Además, desactivamos el elitismo para que el rendimiento de los

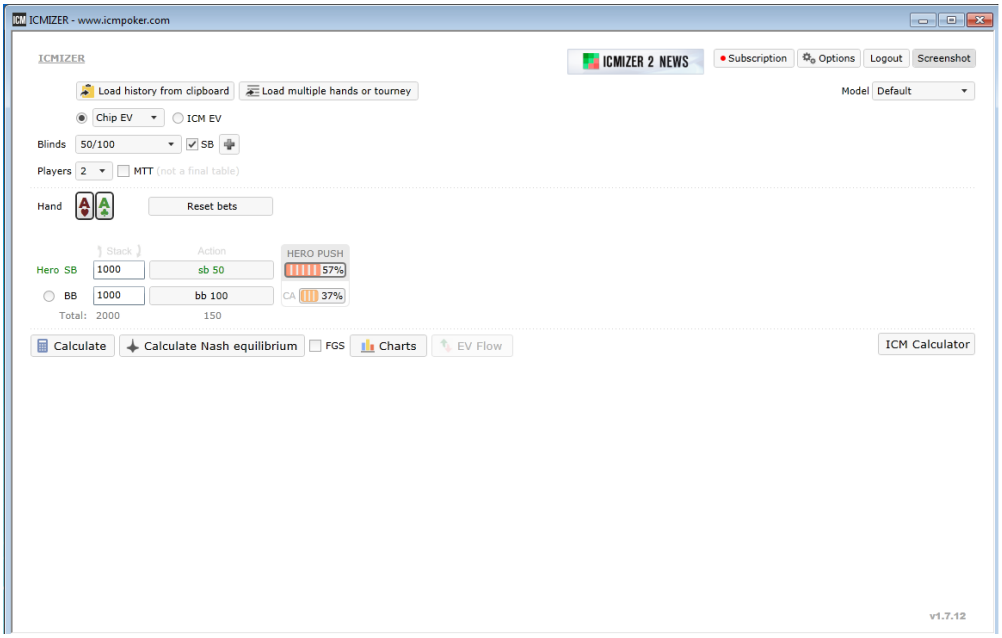


Figura 4.3: Solución de equilibrio para *cap* de 10 ciegas

Experimento	Solución de equilibrio para jugadores simples
Jugadores	Jugadores simples
Tipo de torneo	Round Robin
Elitismo	No
Terminación	Por convergencia del mejor jugador
Gráficas	Nube de puntos

Tabla 4.1: Parámetros de configuración del experimento para la solución de equilibrio de jugadores simples

mejores jugadores sea conforme a la población de ese momento y no a anteriores poblaciones. Por último, el algoritmo termina cuando la población converge a la solución de equilibrio. Los parámetros de configuración quedan resumidos en la Tabla 4.1.

4.2.1. Resultados obtenidos

En cada una de las repeticiones del experimento hemos recogido el número de jugadores de la población inicial, el tiempo que ha tardado en converger la población y la solución obtenida. A su vez, hemos comparado estas soluciones con la solución de equilibrio canónica 57/37, determinando su rendimiento relativo, para dar una medida de lo buena que es la solución resultado comparado con la que a priori sabemos es la óptima (véase Tabla 4.2).

El experimento se ha repetido para tamaños de población de entre 20 y 40 jugadores, obteniendo en cada uno de ellos soluciones muy parecidas a la solución canónica. Como podemos observar, las soluciones sitúan el porcentaje de *raise* y

N	t (min)	Solución	Rendimiento relativo (bb/hands)
20	5.08	50.45/40.56	-0.19
20	10.21	56.55/42.40	-0.67
20	6.16	53.32/38.63	0.64
20	7.55	56.27/34.81	-1.16
25	11.91	55.56/37.18	0.03
25	10.81	58.36/39.77	-0.05
30	14.21	54.49/43.58	-1.26
30	17.16	59.93/39.59	1.16
40	32.41	58.41/35.02	0.52

Tabla 4.2: Resultados del experimento solución de equilibrio con jugadores simples

de *call* en torno al 57 % y al 37 % respectivamente, como es de esperar, y muestran un rendimiento relativo de entre 0 y 1 bb/hands aproximadamente de diferencia, lo que demuestra la calidad de las mismas. Recordamos que una diferencia de menos de 5bb/hands entre el rendimiento de dos estrategias no es significativa y, a efectos prácticos, pueden considerarse igual de eficaces. Podemos notar también la complejidad cuadrática del algoritmo conforme aumentamos el número de jugadores involucrados en la evolución. Además, como se desprende de la tabla, una población inicial de 20 jugadores posee la suficiente diversidad como para alcanzar una solución de equilibrio bastante buena, por lo que no necesitamos utilizar tamaños grandes de población para inferir estas soluciones.

4.3. Modo adaptativo con jugadores simples

El objetivo de este experimento es encontrar una estrategia optimal respecto a otra dada en el marco de los jugadores simples. Enfrentamos a una población de jugadores simples contra un único jugador simple parametrizado de antemano. Aquellos que obtengan mejor rendimiento contra ese jugador concreto son los que mayor probabilidad tienen de evolucionar. De esa forma, a lo largo de las sucesivas generaciones, los jugadores de la población irán aprendiendo estrategias cada vez más eficientes hasta llegar a una estrategia optimal.

Los jugadores evolucionan aplicando el cruce y mutador de este tipo de jugadores. Además, activamos el elitismo para conservar las mejores soluciones de cada generación, preservar el buen material genético y construir sobre él. Por último, el criterio de terminación es por convergencia, es decir, paramos la iteración cuando, durante un cierto número de generaciones, el mejor jugador no es capaz de mejorar su rendimiento contra el oponente y la función objetivo se estabiliza. Los parámetros de configuración quedan resumidos en la Tabla 4.3.

4.3.1. Resultados obtenidos

Para probar el modo adaptativo, hemos analizado dos casos en los que los oponentes se salen de la solución de equilibrio para jugar con una estrategia agresiva

Experimento	Modo adaptativo para jugadores simples
Jugadores	Jugadores simples
Tipo de torneo	Todos contra uno
Elitismo	Si
Terminación	Por convergencia del mejor jugador
Gráficas	Nube de puntos

Tabla 4.3: Parámetros de configuración del experimento para el modo adaptativo de jugadores simples

N	**Oponente**	**t (min)**	**Solución**	**Rendimiento relativo (bb/hands)**
20	70/60	1.18	45.33/46.97	11.08
20	70/60	1.25	47.21/49.43	14.18
25	70/60	1.38	42.36/42.45	15.77
25	70/60	1.36	52.42/54.32	15.25
30	70/60	2.23	49.86/49.86	15.25
30	70/60	2.26	43.36/49.46	15.10
20	40/30	1.93	99.75/31.52	6.01
20	40/30	1.43	97.42/29.33	6.50
25	40/30	2.16	99.30/27.78	7.06
25	40/30	1.91	90.03/27.42	6.80
30	40/30	1.91	94.84/22.29	5.98
30	40/30	2.43	85.42/31.52	6.13

Tabla 4.4: Resultados del experimento solución de equilibrio con jugadores simples

por un lado y conservadora por el otro (véase Tabla 4.4). En el primer caso, el oponente agresivo hace *raise* con el 70 % de las mejores manos y *call* con el 60 %. Podemos observar que nuestros jugadores evolucionan a estrategias que oscilan en torno al 45/45 de *raise/call*, obteniendo un rendimiento relativo respecto al oponente de entre 10 y 15 bb/hands y que constituyen las estrategias que mejor se comportan frente a este. En el segundo caso, el oponente conservador hace *raise* con el 40 % de las mejores manos y *call* con el 30 %. En este caso, la evolución de los jugadores encuentra estrategias claramente mejores, como demuestra su rendimiento relativo, en torno al 95/25 de *raise/call*. Nuevamente, conviene notar el carácter lineal del algoritmo empleado, que se traduce en tiempos muy manejables.

4.4. Solución de equilibrio con jugadores complejos

Este experimento es similar al realizado para los jugadores simples. En este caso, evolucionamos a una población de jugadores complejos, usando torneos de tipo todos contra todos como medida del rendimiento, hasta que alcanzan una solución de equilibrio. Nuevamente, aquel jugador que juegue en equilibrio es el que mejor rendimiento global tendrá contra una población de jugadores variada y

Experimento	Solución de equilibrio para jugadores complejos
Jugadores	Jugadores complejos
Tipo de torneo	Todos contra uno
Elitismo	No
Terminación	Por convergencia del mejor jugador
Gráficas	Mapas de calor del mejor jugador y de la población global

Tabla 4.5: Parámetros de configuración del experimento para la solución de equilibrio de jugadores complejos

es el que sobrevivirá a lo largo de las generaciones. Como antes, el inicializador, cruce y mutador son los específicos para este jugador en concreto. Además, no activamos el elitismo y la terminación es por convergencia. Los parámetros de configuración quedan resumidos en la Tabla 4.5.

4.4.1. Resultados obtenidos

En los resultados anteriores hemos observado que una población de entre 20 y 30 de jugadores simples proporcionaba soluciones de equilibrio interesantes en tiempos relativamente razonables. En este experimento, dada la mayor complejidad de los jugadores involucrados, el tiempo de ejecución se ve incrementado de forma significativa.

La ejecución que aquí analizamos ha sido configurada con una población de 30 jugadores y ha convergido a la solución final en un tiempo de 45 min. (véase Figura 4.4). Debido a los inicializadores que hemos empleado, los jugadores empiezan con estrategias basadas en las de los jugadores simples, en las que se apuesta con un porcentaje determinado de las mejores manos. Observamos que esta estrategia se estabiliza rápidamente, en las primeras generaciones, para el mejor jugador de la población en un valor en torno al 57/37 de la solución de equilibrio; en el caso de la ejecución que mostramos, la solución se estabiliza en un 53/32, que es una solución aceptable en términos de rendimiento, como comprobamos en los experimentos para jugadores simples. Las sucesivas iteraciones del algoritmo tratan de salirse de esa característica regla de apostar con las mejores manos, lo que se aprecia en el gráfico de la población global (véase Apéndice B), sin encontrar una estrategia que mejore la del mejor jugador, el cuál permanece prácticamente estable, salvo pequeñas variaciones insustanciales, durante gran parte de la ejecución. De este modo, este experimento nos permite asegurar que la estrategia basada en apostar en función de un porcentaje de las mejores manos es acertada.

4.5. Modo adaptativo con jugadores complejos

Al igual que en el modo adaptativo para jugadores simples, enfrentamos a una población de jugadores complejos contra un único jugador complejo parametrizado de antemano. Los jugadores más fuertes evolucionarán a lo largo de las

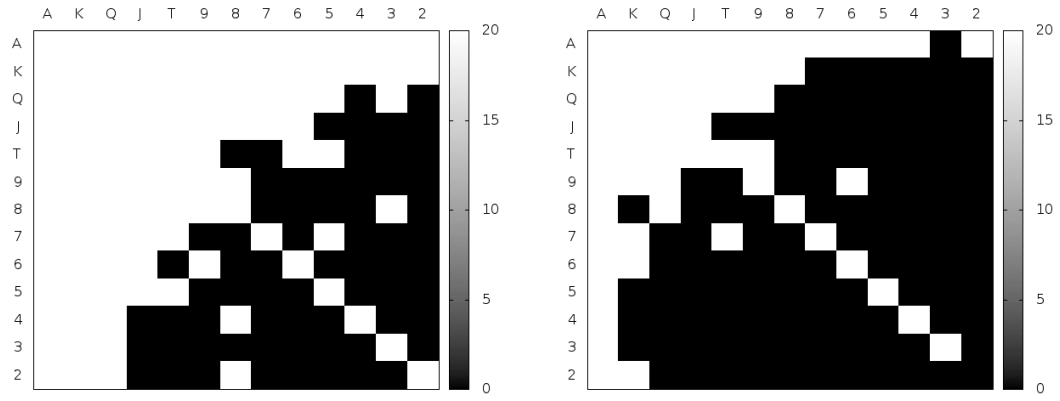


Figura 4.4: Solución de equilibrio para jugadores complejos. A la izquierda la tabla de *raise*; a la derecha la tabla de *call*

Experimento	Modo adaptativo para jugadores complejos
Jugadores	Jugadores complejos
Tipo de torneo	Todos contra uno
Elitismo	Si
Terminación	Por convergencia del mejor jugador
Gráficas	Mapas de calor para el mejor jugador y la población global

Tabla 4.6: Parámetros de configuración del experimento para el modo adaptativo de jugadores complejos

generaciones hasta dar con una estrategia optimal.

En cuanto a los parámetros de configuración, activaremos el elitismo para conservar el buen material genético de una generación a otra, el inicializador, cruce y mutador son los propios de este jugador, y la terminación es por convergencia. Resumimos los parámetros de configuración en la Tabla 4.6.

Cabe destacar que para este ejemplo es especialmente importante activar el elitismo pues, al haber muchos más parámetros, el cruce tiene una probabilidad menor de intercambiar el material genético que hace fuerte a una estrategia y podemos acabar con estrategias peores que las de partida. Activando el elitismo, una vez que hemos llegado a una buena estrategia, trabajamos sobre ella sin temor a perderla.

4.5.1. Resultados obtenidos

Del mismo modo que en el experimento homólogo para jugadores simples, hemos enfrentado a una población de jugadores complejos contra una estrategia agresiva 70/60 y una conservadora 40/30. Como en el anterior experimento de equilibrio, las soluciones del mejor jugador se estabilizan rápidamente en soluciones del estilo de las de los jugadores simples y son las siguientes iteraciones las que tratan de explorar soluciones que se alejan de estas sin éxito alguno. Mostramos

en la Figura 4.5 el resultado de la ejecución del algoritmo para el primer caso, donde se ha utilizado una población de 30 jugadores y se ha empleado un tiempo de 2.01 min, resultando en una estrategia que ciertamente está en torno al 45/45 buscado de los experimentos anteriores. En la Figura 4.6 mostramos el resultado de la ejecución para la estrategia conservadora, donde se ha empleado una población de 30 jugadores con un tiempo de 1.95 min y donde se ha obtenido una solución que, como esperábamos, es cercana a un 90/25.

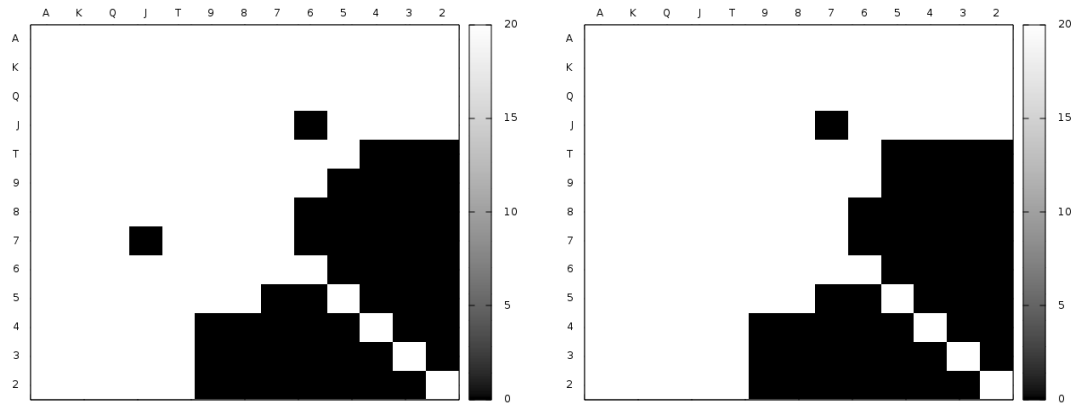


Figura 4.5: Estrategia optimal contra un jugador 70/60. A la izquierda la tabla de *raise*; a la derecha la tabla de *call*

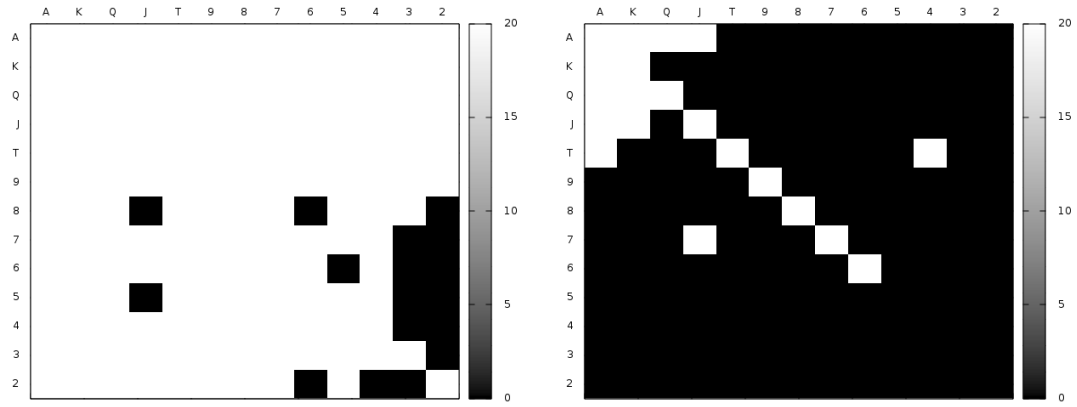


Figura 4.6: Estrategia optimal contra un jugador 40/30. A la izquierda la tabla de *raise*; a la derecha la tabla de *call*

4.6. Solución de equilibrio con jugadores complejos en un escenario de stack efectivo variable

A diferencia de los experimentos anteriores, en este trabajaremos en el mismo escenario pero variando la cantidad máxima apostable. Tenemos que recurrir,

Experimento	Solución de equilibrio para jugadores complejos con cantidad máxima apostable variable
Jugadores	Jugadores complejos con cantidad máxima apostable
Tipo de torneo	Todos contra todos
Elitismo	No
Terminación	Por convergencia del mejor jugador
Gráficas	Mapas de calor para el mejor jugador y la población global

Tabla 4.7: Parámetros de configuración del experimento para la solución de equilibrio de jugadores complejos con catidad máxima apostable variable

entonces, a los jugadores que llevan en la tabla la cantidad máxima a apostar en cada caso.

Los enfrentamos a un torneo de tipo todos contra todos y los evolucionamos hasta encontrar la solución de equilibrio. Debido a la naturaleza de su inicializador, el cual inicializa las manos a la cantidad máxima apostable, las sucesivas generaciones servirán para que el mutador explore otras cantidades máximas apostables. Cuando el mejor jugador de la población no consiga mejorar, a pesar de continuar explorando otras cantidades, su rendimiento habrá convergido y terminaremos la evolución.

Los parámetros de configuración para este experimento están recogidos en la Tabla 4.7.

4.6.1. Resultados obtenidos

Un factor de suma importancia en este experimento ha sido el número de jugadores de la población inicial, por lo que hemos probado tamaños de entre 10 y 50 jugadores hasta encontrar el idóneo. Dada la complejidad cuadrática del algoritmo y la gran magnitud del espacio de jugadores, un número elevado individuos conducía a un menor número de iteraciones por unidad de tiempo y, a la larga, soluciones menos refinadas. Por otro lado, un número reducido de jugadores iniciales resultaba en una convergencia demasiado temprana.

Por ello, para la ejecución analizada hemos elegido un compromiso entre ambos con una población de 20 jugadores, la cual ha tardado un tiempo de 3h en converger. Si observamos el proceso de evolución (véase Apéndice C) podemos comprobar como la solución del mejor toma rápidamente la forma de una estrategia de equilibrio de jugadores simples, como sucedía con los jugadores anteriores. A lo largo de las siguientes generaciones, las soluciones van refinando la cantidad máxima apostable en cada celda hasta alcanzar el equilibrio en la población. Debido al alto grado de granularidad en el espacio de jugadores, existen multitud de óptimos locales que el algoritmo puede alcanzar, resultando en muy diversas soluciones en cada ejecución. Aun así, las solución obtenida (véase Figura 4.7) fue enfrentada a la solución de equilibrio 57/37 de los jugadores simples, en este otro escenario, resultando la estrategia obtenida ligeramente más ventajosa con

un rendimiento relativo de 2.76 bb/hands.

Por último, en este experimento tratábamos de encontrar la solución de equilibrio según la clasificación de Slamsky-Chubukov, no siendo posible debido a la orografía rica en óptimos locales del espacio de estrategias. No obstante, enfrentada contra esta última estrategia, observamos que su comportamiento en términos de rendimiento es significativamente parecido, con un rendimiento relativo de 0.54 bb/hands a favor de la solución de equilibrio de Slamsky-Chubukov.

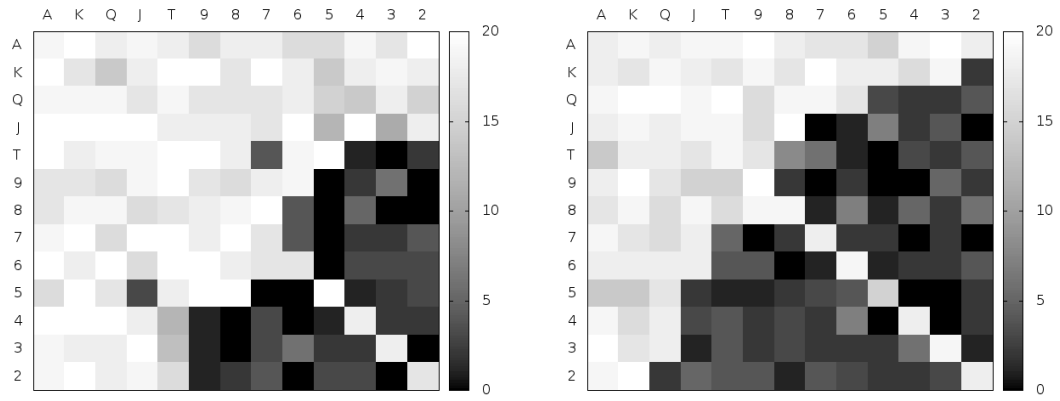


Figura 4.7: Solución de equlilibrio para jugadores complejos con cantidad máxima apostable variable

4.7. Conclusiones

Como hemos podido comprobar, los algoritmos genéticos se comportan bien y encuentran las mejores soluciones cuando la dificultad del espacio de jugadores es relativamente reducida. Conforme aumentamos la complejidad de las decisiones que tienen que tomar, aumenta la magnitud del espacio de jugadores asociado y se hace más acuciante la necesidad de utilizar inicializadores, cruces y mutadores que hagan una exploración *inteligente* del espacio de jugadores. Aun así, dada su extensión, tienen dificultades en encontrar las soluciones óptimas y tienden a concentrarse en óptimos locales.

Además, el coste algorítmico cuadrático de los experimentos de equilibrio ha sido prohibitivo, no permitiendo utilizar un número suficientemente grande de jugadores en cada caso y limitando la potencia a la hora de explorar el espacio de jugadores.

Apéndice A

Evolución de jugadores simples

En este apéndice mostramos una evolución de jugadores simples. El experimento asociado es *Modo adaptativo para jugadores simples* y en él tratamos de encontrar una estrategia optimal contra una estrategia 70/60 de *raise/call*. Mostramos la evolución de la población cada 2 iteraciones aproximadamente (véase Figura A.1).

Podemos observar cómo la población, que inicialmente está uniformemente distribuida por todo el plano, se va concentrando en torno a la solución final 45/45 conforme avanza la evolución. Cabe destacar, en la última iteración, cómo son varios, y no una, los individuos que han sobrevivido. Esto es así porque, entre estas soluciones, la diferencia de rendimiento es poco significativa y, a efectos prácticos, son muy parecidas. De entre ellas, el algoritmo genético escoge la que mayor rendimiento haya mostrado en esa iteración.

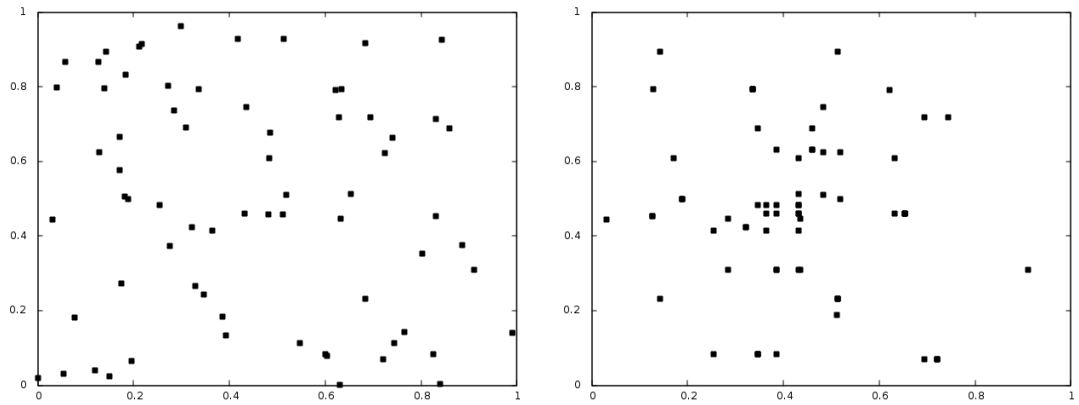
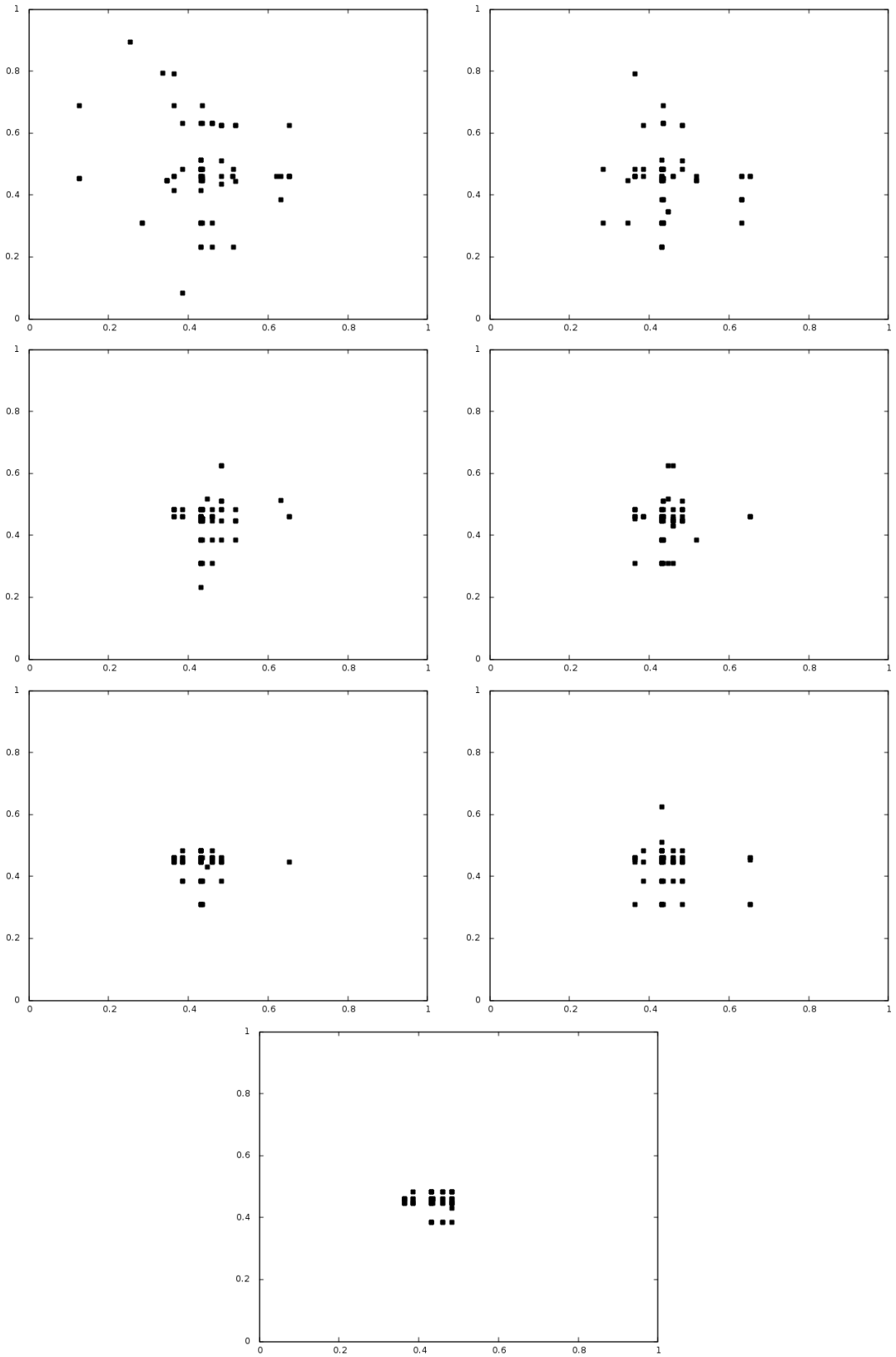


Figura A.1: Evolución de jugadores simples


0
0.2
0.4
0.6
0.8
1

Apéndice B

Evolución de jugadores complejos

En este apéndice mostramos un ejemplo de evolución de una población de 30 jugadores complejos hasta el equilibrio. El experimento asociado es, por tanto, *Solución de equilibrio para jugadores complejos*. En la Figura B.1 está representada la evolución de la tabla *raise* del mejor jugador en momentos separados por aproximadamente 7 iteraciones, mientras que en la Figura B.1 está representado el estado de la población global en cada uno de los momentos anteriores.

Cabe destacar, como comentamos en la sección 4.4.1, cómo la solución del mejor jugador adquiere rápidamente, en las primeras generaciones, una estrategia basada en la estrategia de equilibrio de los jugadores simples, en la que se apuesta con un porcentaje en torno al 57/37 de las mejores manos. Esta solución se mantiene estable prácticamente durante toda la evolución, con pequeñas modificaciones que resultan insustanciales de cara al rendimiento. En las sucesivas iteraciones, se aprecia en los mapas de calor de la población global cómo el algoritmo genético trata de salirse de esta particular estrategia, con cambios más agresivos conforme avanza la población sin lograr una estrategia mejor.

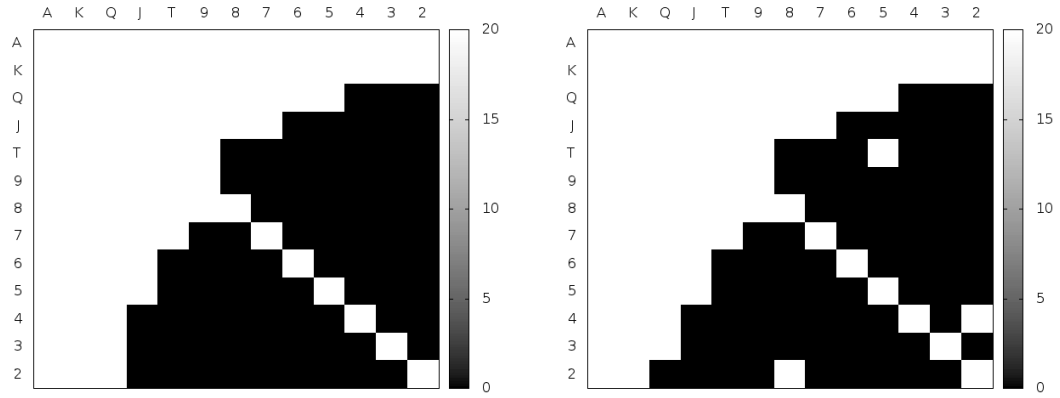
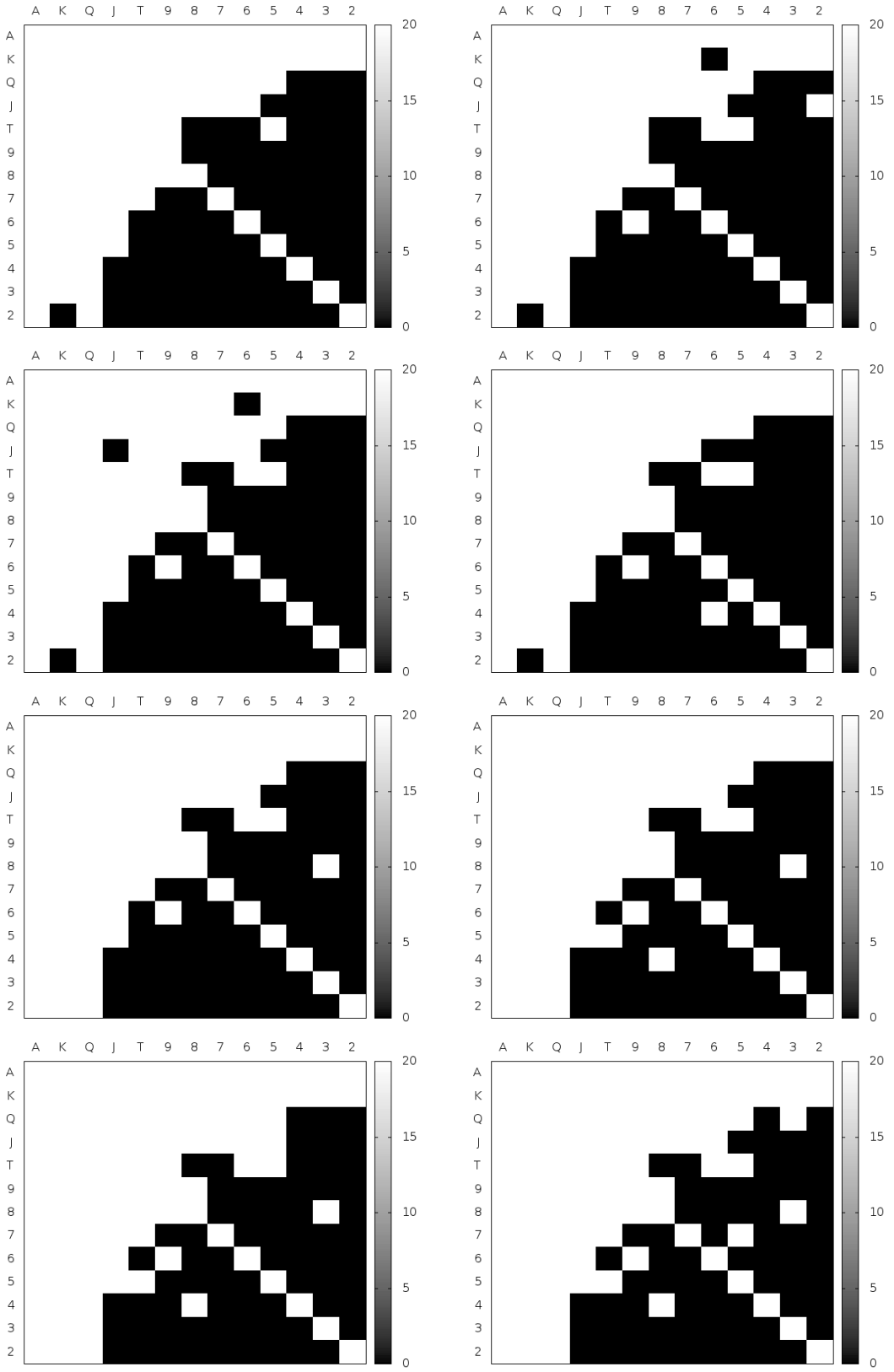


Figura B.1: Mapa de calor de la tabla *raise* del mejor jugador


A K Q J T 9 8 7 6 5 4 3 2
A K Q J T 9 8 7 6 5 4 3 2
20 15 10 5 0

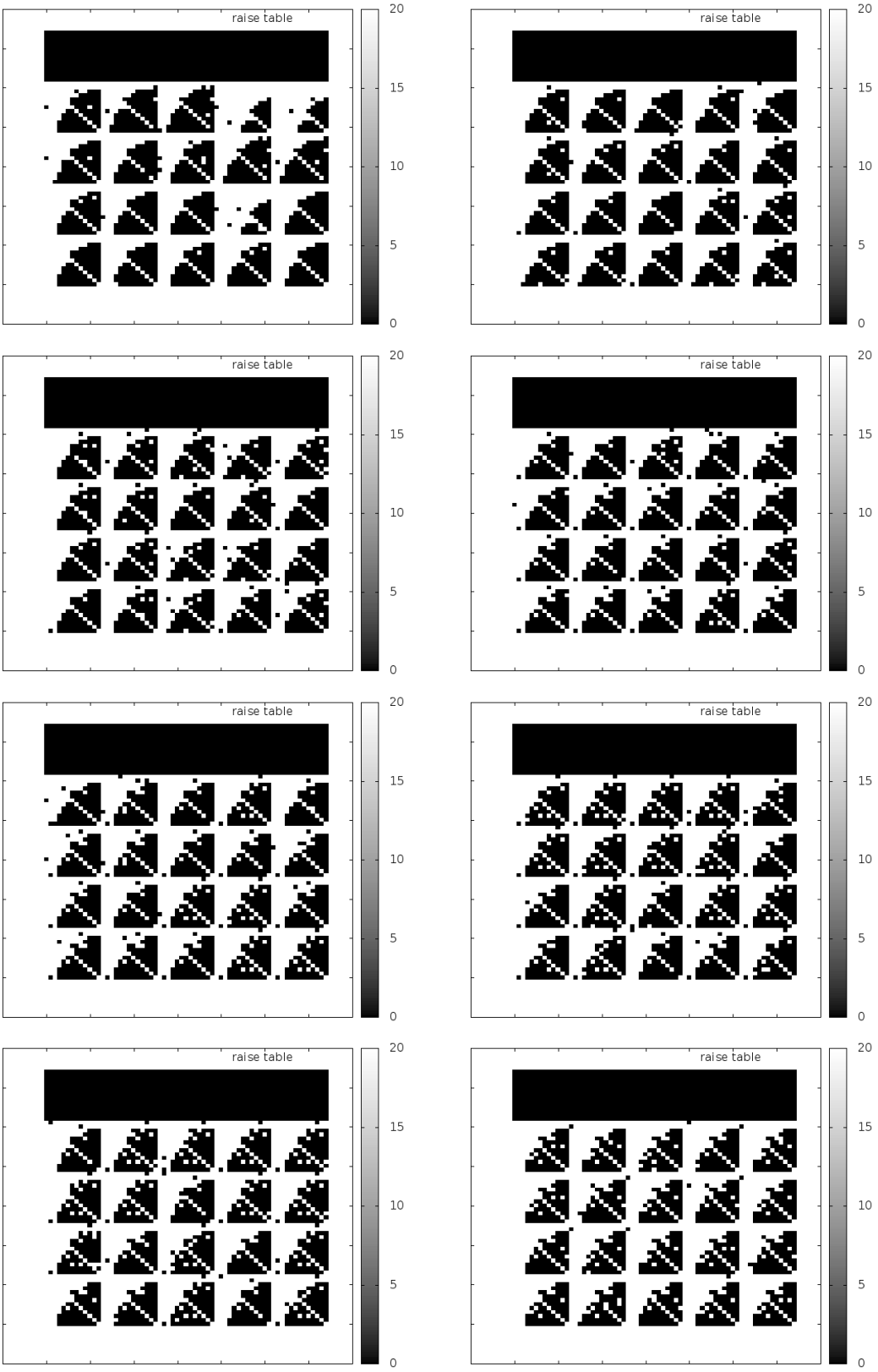
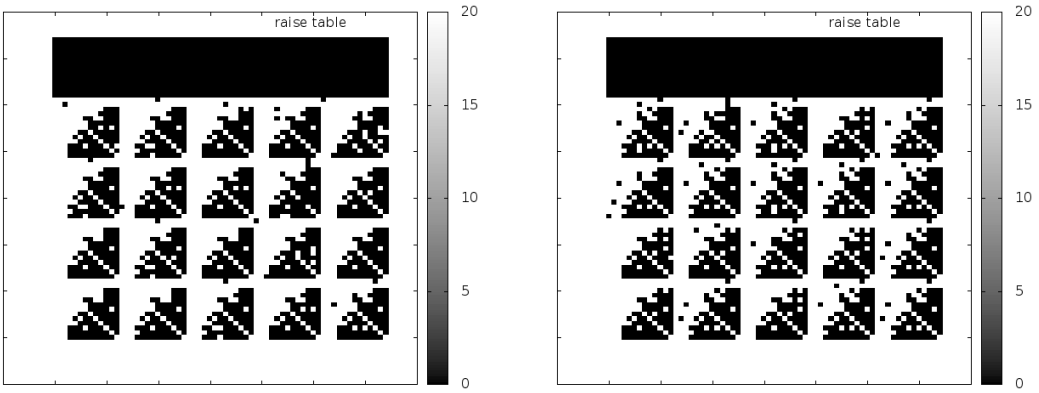


Figura B.1: Mapa de calor de la tabla *call* de la población total


raise table
20
15
10
5
0
raise table
20
15
10
5
0

Apéndice C

Evolución de jugadores complejos con stack efectivo variable

En este apéndice se muestra un ejemplo de evolución de una población de 20 jugadores complejos en un escenario en el que el stack efectivo varía con cada mano. En las Figuras C.1 y C.0 están representados los mapas de calor de la tabla *call* del mejor jugador y de la población total, respectivamente, en intervalos separados por aproximadamente 5 iteraciones. Obsérvese cómo la solución del mejor jugador se adecúa rápidamente, en las primeras generaciones, a la estrategia de equilibrio que siguen los jugadores simples como consecuencia de los inicializadores empleados. En las sucesivas iteraciones, la cantidad máxima apostable en cada celda se va refinando hasta que la población converge.

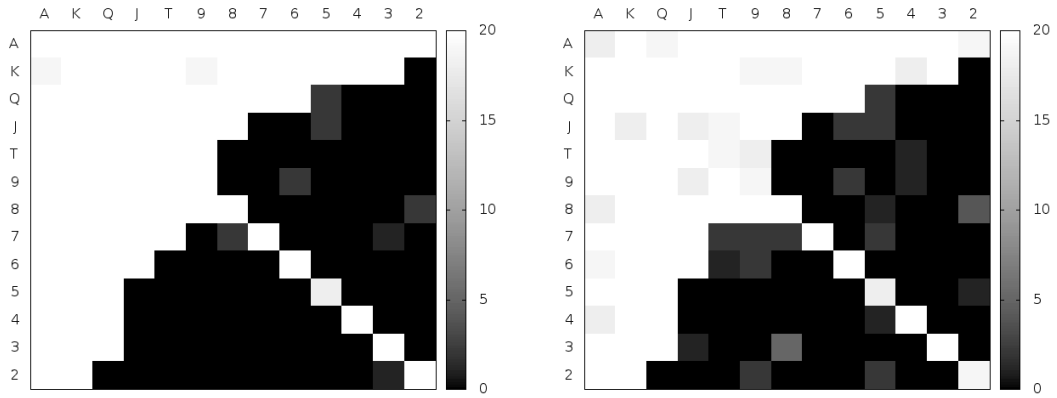
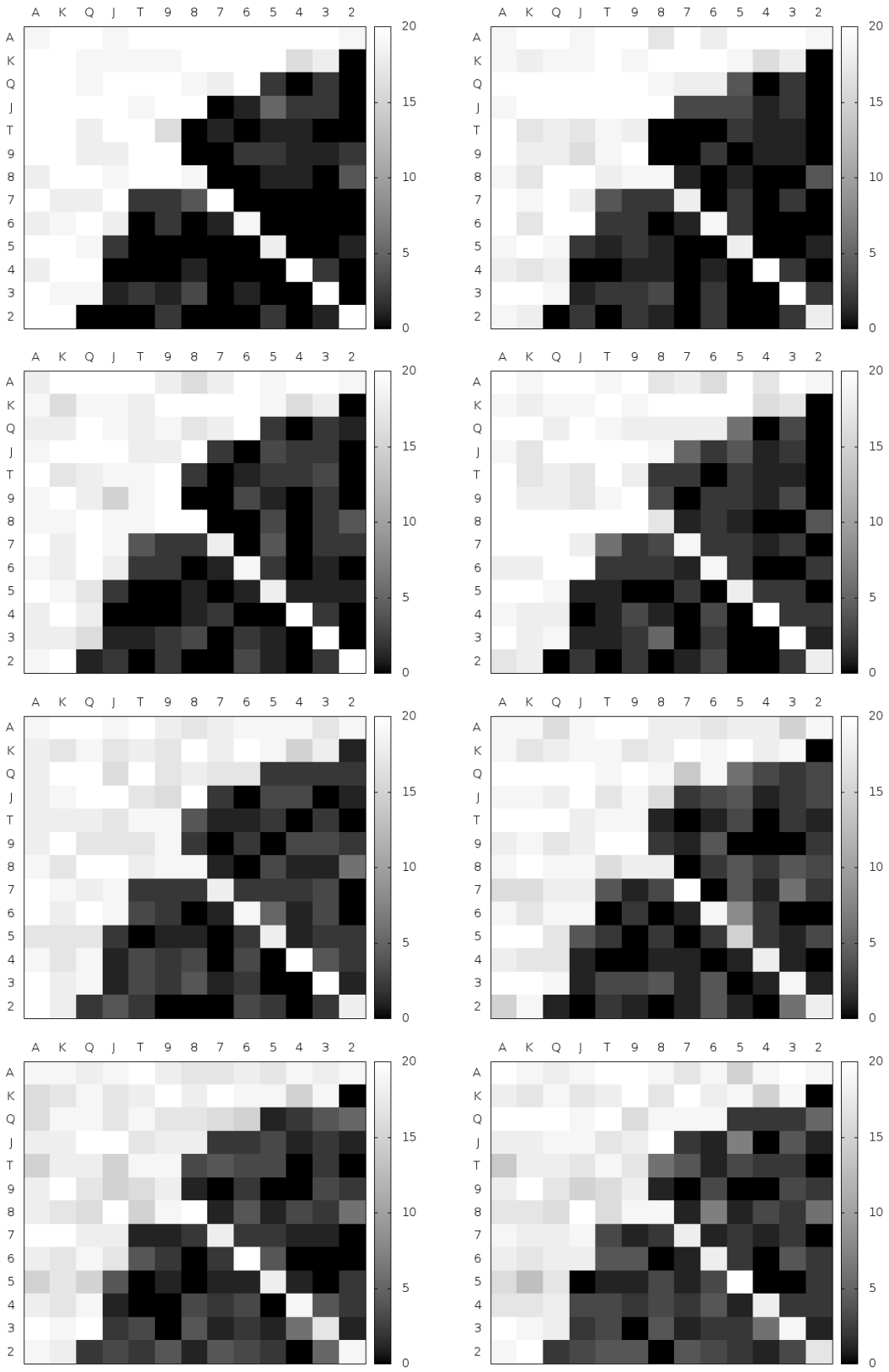
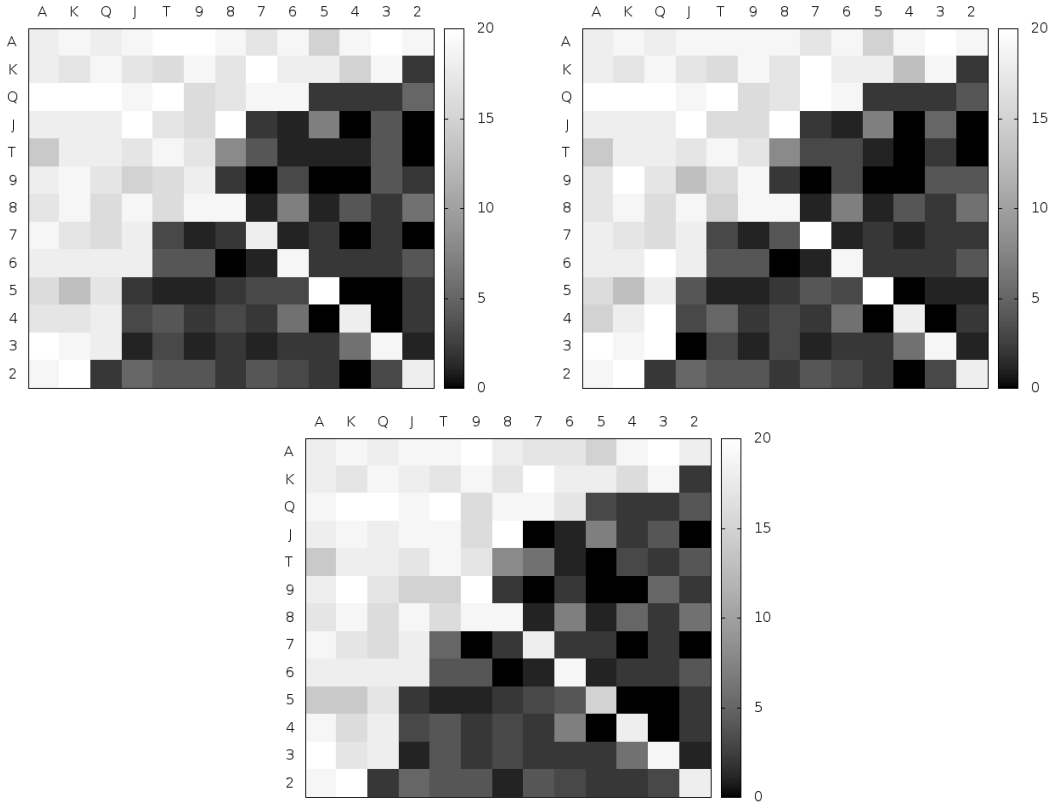


Figura C.1: Mapa de calor de la tabla *call* del mejor jugador


A K Q J T 9 8 7 6 5 4 3 2
20
15
10
5
0



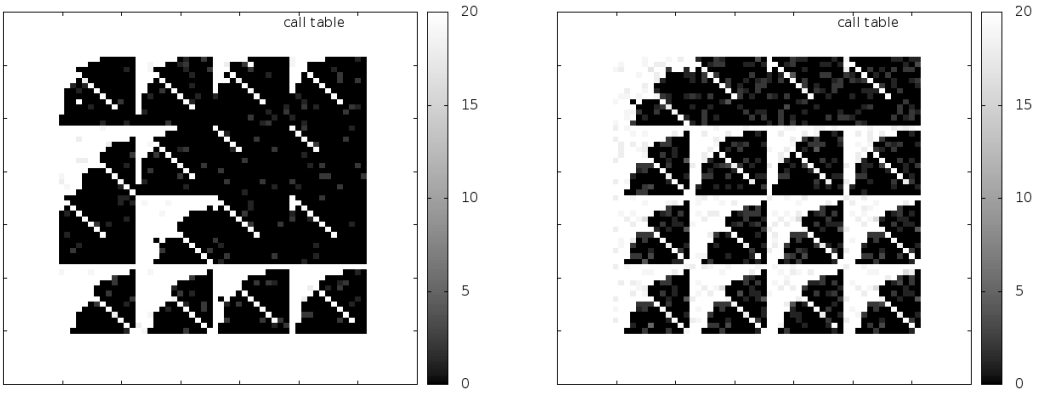
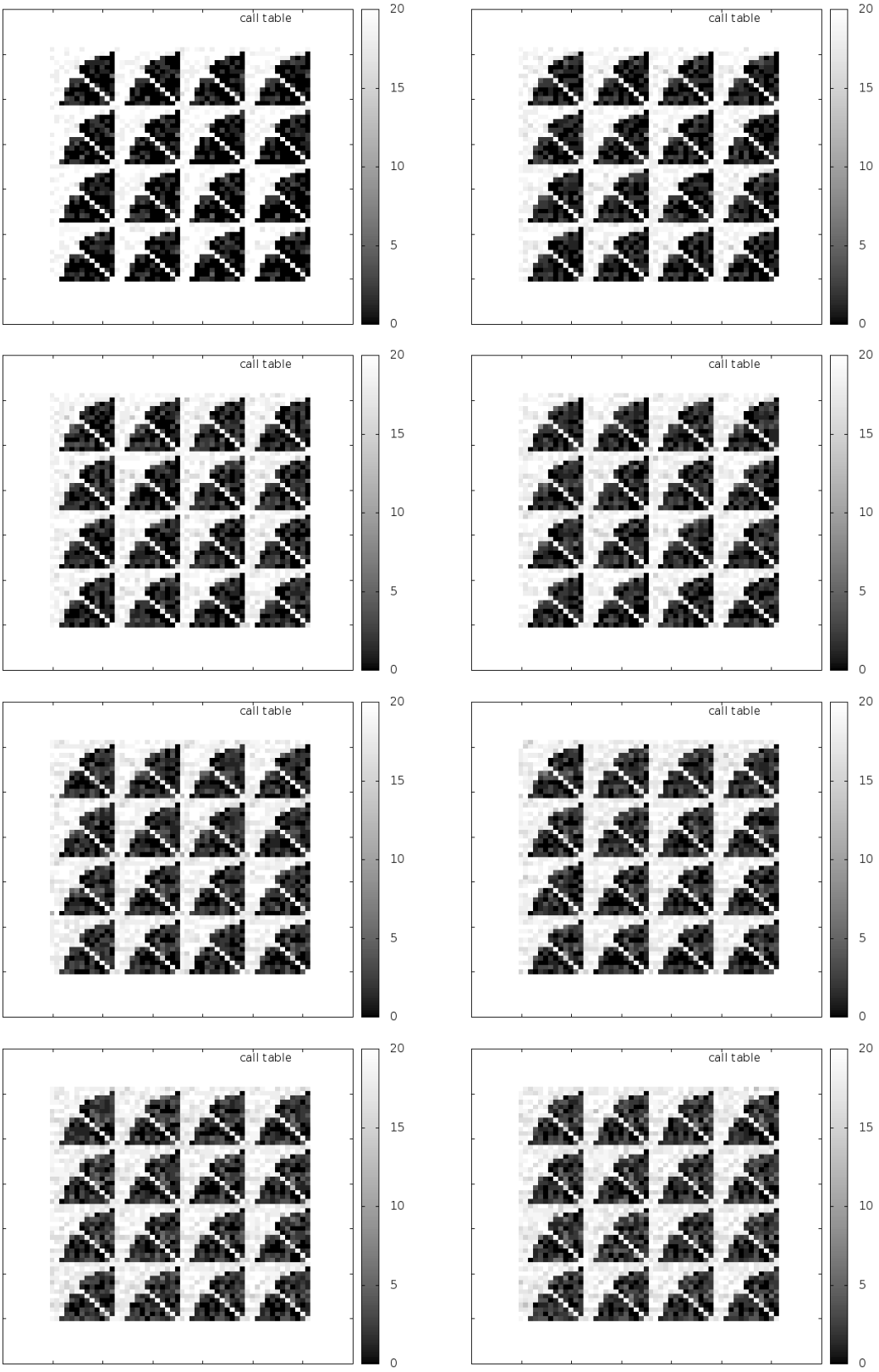
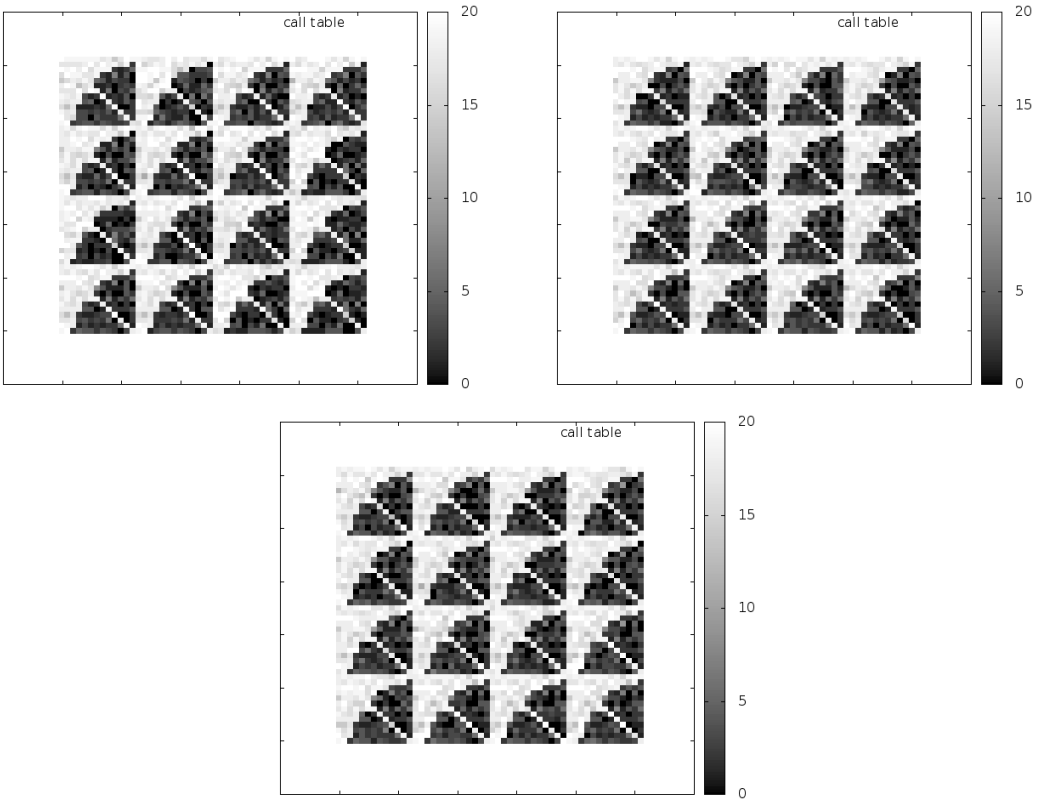


Figura C.0: Mapa de calor de la tabla *call* de la población total


call table
20
15
10
5
0


call table
20
15
10
5
0
call table
20
15
10
5
0
call table
20
15
10
5
0

Bibliografía

[1] Brains Vs. AI | Carnegie Mellon School of Computer Science. https://www.cs.cmu.edu/brains-vs-ai.

[2] Gnuplot graphing utility. http://www.gnuplot.info/.

[3] T. Bakker. *Analytical No-Limit Hold'em: Crushing Mid-Stakes Short-Handed Games*. Two Plus Two Publishing, LLC, Nov. 2010.

[4] M. Bowling, N. Burch, M. Johanson, and O. Tammelin. Heads-up limit hold'em poker is solved. *Science*, 347, Jan. 2015.

[5] O. C. Derby. Pbots_calc c library: A ranged equity calculator for mit pokerbots. https://github.com/mitpokerbots/pbots_calc.

[6] T. S. L. D. Michael Maurer, Brian Goetz. Poker-eval c library. https://github.com/v2k/poker-eval.

[7] C. Moshman and D. Zare. *The Math of Hold'em*. Dimat Enterprises, Incorporated, May 2011.

[8] M. Wall. GAlib: Matthew's C++ Genetic Algorithms Library. http://lancet.mit.edu/galib-2.4/.