ITESM Campus Estado de México

Materia: Sistemas Inteligentes

Fernando Gómez Herrera <A01020319>
Rodolfo Andrés Ramírez Valenzuela <A01169701>
Jonathan Josué Patlán Reyna <A01372223>

ACTIVIDAD DE PROGRAMACIÓN 3 – ALGORITMOS GENÉTICOS

1. PROBLEMA DEL VIAJERO (TRAVELING SALESMAN PROBLEM)

2. PROBLEMA DE LA MOCHILA (KNAPSACK PROBLEM)

3. ENCONTRAR LA MEJOR MANO PARA UNA PARTIDA DE PÓKER

INTRODUCCIÓN

Los algoritmos genéticos son algoritmos buscan buenas soluciones a un problema de entre un gran número de posibles soluciones. Estos paradigmas computacionales nosotros fueron inspirados en la mecánica de la evolución natural, incluida la supervivencia del más apto, la reproducción y mutación. Esta mecánica es muy adecuadas para resolver una variedad de problemas prácticos , incluyendo problemas computacionales , en muchos campos. Algunas aplicaciones son la optimización, programación automática , aprendizaje automático, la economía, genética de la población, entre otros.

Con un algoritmo genético se comienza con un conjunto de soluciones candidatas (cromosomas) llamadas población y una nueva población se crea a partir de soluciones de una la población "vieja" en la esperanza de conseguir una mejor población. Las soluciones que se eligen para tener nuevas soluciones (descendencia) se seleccionan de acuerdo a su condición física y el proceso se repite hasta que se satisface una cierta condición.

PROBLEMA DEL VIAJERO (TRAVELING SALESMAN PROBLEM)

El TSP es problema de optimización NP-difícil que ha sido ampliamente estudiado en que dados las ciudades y las distancias (costos) entre cada par de ciudades se tiene como objetivo encontrar un recorrido de costo mínimo que visita cada ciudad exactamente una vez.

ARQUITECTURA DE SOFTWARE:

Para la solución del problema se utilizará:

- Matlab R2016a

METODOLOGÍA

Resolver este problema requirió definir los parámetros necesarios para realizar las fases de *reproducción, crossover y mutación*.

A continuación se presenta la forma en que se representaron estas etapas en el algoritmo.

Se tomará un ejemplo de cuatro ciudades, numeradas del 1 al 4. Con la siguiente matriz de distancias.

$$\begin{pmatrix} 0 & 25 & 48 & 33 \\ 25 & 0 & 63 & 52 \\ 48 & 63 & 0 & 19 \\ 33 & 52 & 19 & 0 \end{pmatrix}$$

SOLUCIÓN DEL PROBLEMA:

I. ALFABETO UTILIZADO PARA EL CROMOSOMA.

Alfabeto numérico que contiene a los números naturales.

$$\mathbb{N} = \{ 1,2,3, \dots \}$$

II. LONGITUD DEL CROMOSOMA.

Cada cromosoma representa la serie de ciudades a visitar, en orden. Es decir, la ruta a seguir.

$$P = [C_1 C_2 C_3 \dots C_n]$$

Ejemplos.

P1 = 1234
P2 = 4231
P3 = 1342

III. FUNCIÓN DE "FITNESS".

Debido a que en algoritmos genéticos se busca **maximizar** esta función, no podemos utilizar directamente una función que nos devuelva la distancia recorrida. De ser así estaríamos encontrando la **peor solución al problema**, cuando nuestro objetivo es minimizar dicha distancia. Es por eso que para la función de *fitness* se utilizó la inversa de la distancia recorrida.

Siendo P un cromosoma que representa la ruta a visitar y D(P) la distancia recorrida en esa ruta, la función de fitness se define de la siguiente forma.

$$P = cromosma\ ;\ \ e.g.P = [1234]$$

$$D(P) = \sum_{i=0}^{n} (P_i)$$

$$f(x) = D(P)^{-1} = \frac{1}{D(P)} = \frac{1}{\sum_{i=0}^{n}(P_i)}$$

IV. PORCENTAJE DE MUTACIÓN.

Para el proceso de mutación se utilizaron las siguientes estrategias:

1. Por cada gen en cada cromosoma se genera una probabilidad aleatoria y si ésta es menor a **0.065**, entonces se hace un **intercambio por otro gen aleatorio.**
2. Por cada cromosoma se genera una probabilidad aleatoria y si ésta es menor a **0.024**, entonces ocurre lo siguiente:
    a. Generar un punto aleatorio **P** dentro del Cromosoma.
    b. Se intercambian las posiciones de $C_1 \ldots C_P$ por $C_{P+1} \ldots C_n$ y viceversa.
3. Se emplea **elitismo** y se conserva el mejor camino encontrado reemplazando a un elemento aleatorio de la nueva población.

V. TAMAÑO DE LA POBLACIÓN.

Generación 1

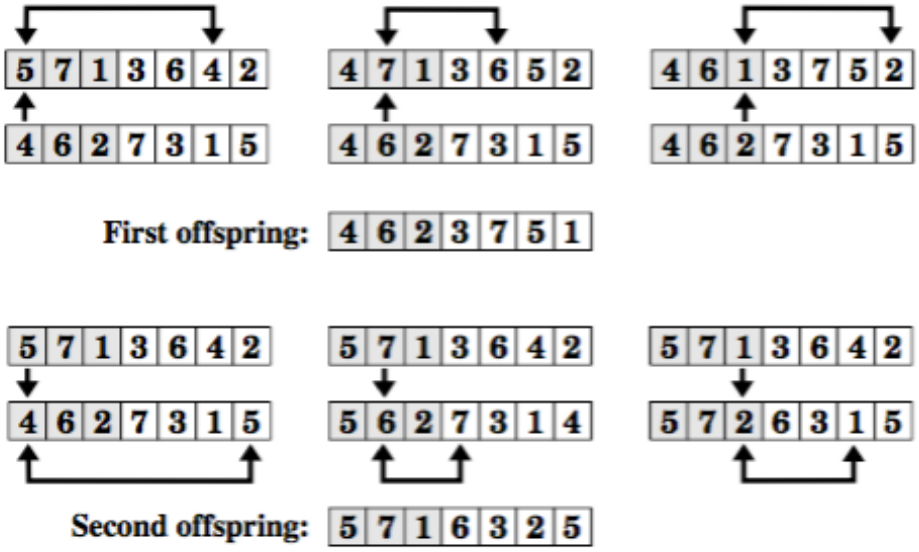
No	Cromosoma	Distance	f(x)	P.Select	E. Count	A. Count
1	1234	62	0.0161	0.270	1.079	2
2	1243	66	0.0152	0.255	1.018	1
3	4312	74	0.0135	0.226	0.905	0
4	3412	67	0.0149	0.250	0.998	1
		SUM	0.0597	1.000	4.000	4
		AVG	0.014925	0.250	1.000	1
		MAX	0.0161	0.270	1.079	2

VI. EJEMPLOS DE POBLACIONES ANTES DE LA REPRODUCCIÓN, AL REPRODUCIRSE, AL REALIZAR EL CROSSOVER Y AL REALIZAR LA MUTACIÓN.

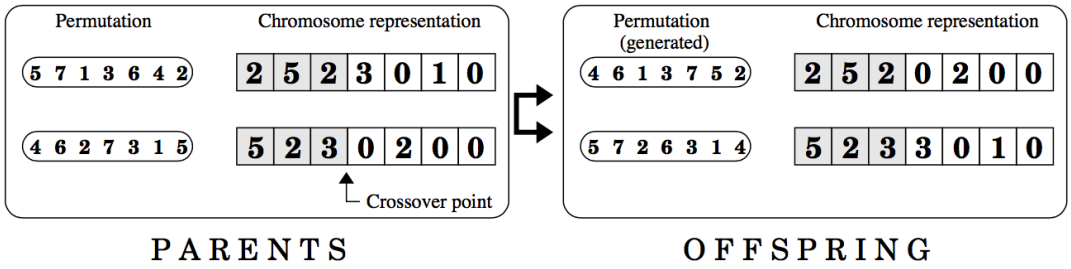
Crossover

Para el crossover se empleó el operador PMX mencionado en el *paper* de *Genetic Algorithm Solution of the TSP Avoiding Special Crossover and Mutation* escrito por *Gokturk Ucoluk*. (Metuedutr, 2016)

La forma en la que funciona se ilustra de la siguiente manera (extraído del paper):



Cruce ordinario que genera dos descendientes de ellos:



PROBLEMA DE LA MOCHILA (KNAPSACK PROBLEM)

El problema de la mochila es muy fácil de explicar. Se le presentará n objetos que pesan cada uno con un peso, por lo general diferente cantidad y se tienen que seleccionar un grupo de objetos que más se aproxima a el peso total de la mochila, pero sin exceder el límite.

ARQUITECTURA DE SOFTWARE:

Para la solución del problema se utilizará:

- Python 3.0

SOLUCIÓN DEL PROBLEMA:

REPRESENTACIÓN DE LOS ELEMENTOS:

Utilizamos una estructura de datos , llamada celda , con dos campos (beneficio y volumen) para representar cada artículo . Luego se utiliza una matriz de tipo celda para almacenar todos los elementos que contiene como se muestra a continuación:

0	1	2	3
20 | 30	5 | 10	10 | 20	40 | 50

I. LONGITUD DEL CROMOSOMA.

Un cromosoma se puede representar en una matriz que tiene un tamaño igual al número de los artículos (en nuestro ejemplo de tamaño 4) .

II. ALFABETO UTILIZADO PARA EL CROMOSOMA.

Alfabeto numérico que contiene a los números naturales.

$$\mathbb{N} = \{ 1,2,3, \dots \}$$

Cada elemento de esta matriz indica si un elemento está incluido en la mochila ("1") o no (' 0 ') . Por ejemplo, el siguiente cromosoma.

0	1	2	3
1	0	0	1

Se indica que el primer y el cuarto artículo están incluidos en la mochila . Para representar a la totalidad población de cromosomas se utiliza una matriz tridimensional.

(cromosomas [tamaño] [número de artículos] [2])

Tamaño representa el número de cromosomas en una población. La segundo dimensión representa el número de elementos que potencialmente pueden ser incluidos en el mochila. La tercera dimensión se utiliza para formar la nueva generación de cromosomas.

III. FUNCIÓN DE "FITNESS".

Calculamos el fitness de cada cromosoma mediante la suma de los beneficios de los elementos que se incluyen en la mochila , mientras se asegura de que no se exceda la capacidad de la mochila . Si el volumen del cromosoma es mayor que la capacidad de la mochila a continuación, uno de los bits en el cromosoma cuyo valor es ' 1 ' se invierte y el cromosoma se comprueba de nuevo .

IV. PORCENTAJE DE MUTACIÓN.

Este proceso se realiza para evitar que se caiga en un extremo local. Llevamos a cabo la mutación en cada posición de bit del cromosoma con un 0,1 % de probabilidad .

V. TAMAÑO DE LA POBLACIÓN.

La población es de un tamaño de 7, sin embargo se pueden agregar más ítems si se requiere.

VI. EJEMPLOS DE POBLACIONES ANTES DE LA REPRODUCCIÓN, AL REPRODUCIRSE, AL REALIZAR EL CROSSOVER Y AL REALIZAR LA MUTACIÓN.

Crossover es el proceso de combinación de los bits de un cromosoma con las de otro. Esto es para crear una descendencia para la próxima generación que hereda rasgos de ambos padres. Crossover elige al azar un locus e intercambia las subsecuencias antes y después de ese locus

entre dos cromosomas para crear dos crías [2] . Por ejemplo , considere los siguientes padres y un punto de cruce en la posición 3 :

Parent 1	1 0 0 | 0 1 1 1
Parent 2	1 1 1 | 1 0 0 0
Offspring 1	1 0 0 1 0 0 0
Offspring 2	1 1 1 0 1 1 1

En este ejemplo , Hijos 1 hereda los bits en la posición 1 , 2 y 3 de la parte izquierda del punto de cruce de los padres 1 y el resto de la parte derecha del punto de cruce de la madre 2. Del mismo modo , Hijos 2 hereda bits de rango 1 , 2 y 3 desde el lado izquierdo del Padre 2 y el resto de el lado derecho de los padres 1.

La mutación se realiza después de cruce para impedir que caigan todas las soluciones en la población en un óptimo local del problema resuelto . La mutación cambia la nueva descendencia por voltear los bits de 1 a 0 o de 0 a 1. La mutación puede ocurrir en cada posición de bit en la cadena con cierta probabilidad , por lo general muy pequeña (por ejemplo 0,001) . Por ejemplo , considere la siguiente cromosoma con mutación puntual en la posición 2 :

Not mutated chromosome:	1 *0* 0 0 1 1 1
Mutated:	1 *1* 0 0 1 1 1

El 0 en la posición 2 voltea a 1 después de la mutación .

El siguiente ejemplo muestra la población inicial y podemos observar crossover con la combinación rojo y verde, y también con la combinación rosa y azul. Los elementos subrayados en color naranja ejemplifican la mutación.

Población 1	Población 2
[2, 2, 2, 0, 1, 1, 1] [2, 2, 0, 2, 1, 0, 0] [2, 1, 2, 0, 2, 0, 2] [0, 1, 1, 2, 0, 2, 1] [1, 2, 0, 2, 1, 0, 0] [2, 0, 1, 0, 1, 2, 0] [2, 2, 0, 1, 2, 2, 2]	[2, 2, 2, 0, 1, 1, 1] [2, 2, 2, 1, 2, 0, 2] [2, 1, 0, 2, 1, 0, 0] [0, 1, 0, 2, 0, 0, 0] [1, 2, 0, 2, 1, 2, 1] [2, 0, 1, 0, 1, 2, 0] [2, 2, 0, 1, 2, 2, 2]

ENCONTRAR LA MEJOR MANO PARA UNA PARTIDA DE PÓKER

El póker es un juego que tiene múltiples variantes hay una cosa que es común a todas las variantes, las manos que un jugador puede obtener y se dividen en los siguientes diez categorías:

Royal Flush	Una combinación de un diez a un as con las cinco cartas del mismo palo. En el póquer todos los juicios se clasifican por igual.
Straight Flush	Cualquier combinación con todas las cartas del mismo palo.
Four of a Kind	Cuatro cartas del mismo valor.
Full House	Tres cartas del mismo valor , junto con cualquier par de cartas del mismo valor.
Flush	Cinco cartas del mismo palo (no consecutivas). La carta más alta de las cinco determina el rango de la descarga.
Straight	Cinco cartas consecutivas de diferente palo.
Three of a Kind	Tres cartas del mismo valor .
Two Pair	Cualquier par de cartas del mismo valor , junto con otras dos cartas del mismo valor.
One Pair	Dos cartas del mismo valor.
High Card	Cualquier mano no en las manos antes mencionados.

ARQUITECTURA DE SOFTWARE:

Para la solución del problema se utilizará:

- Python 3.0

SOLUCIÓN DEL PROBLEMA:

I. ALFABETO UTILIZADO PARA EL CROMOSOMA.

El alfabeto del cromosoma son los números naturales, letras mayúsculas y minúsculas.

II. LONGITUD DEL CROMOSOMA.

Una posible solución puede ser el codificado en un cromosoma único, que es simplemente un conjunto de valores (o genes) que describen dicha solución.

El conjunto de soluciones puede ser codificada en los cromosomas de 5 genes para esta implementación.

III. FUNCIÓN DE "FITNESS".

Representamos tarjetas como enteros de 32 bits, sólo son enteros. La mayoría de los bits son utilizados , y tienen un significado específico:

Card:

```
   bitrank     suit rank   prime
+--------+--------+--------+--------+
|xxxbbbbb|bbbbbbbb|cdhsrrrr|xxpppppp|
+--------+--------+--------+--------+
```

1) p = número primo de rango (deuce=2, Trey=3, cuatro=5, ..., as=41)

2) r = rango de la carta (deuce=0,trey=1,cuatro=2,five=3,...,ace=12)

3) cdhs = suit of card (bit turned on based on suit of card)

4) b = bit activado en función de rango de tarjeta

5) x = no usado

Esta representación nos permitirá hacer cosas muy importantes como :

- Hacer un prodcuto pirmo único para cada mano
- Detectar flushes
- Detectar straights

IV. PORCENTAJE DE MUTACIÓN.

Se establece como probabilidad de mutación 0.01, lo cual indica que solo el 1% de los cromosomas tienden a ser mutados

V. TAMAÑO DE LA POBLACIÓN.

La población contiene seis individuos cada uno con cinco genes, como se muestra a continuación con el siguiente ejemplo:

[3♦], [J♠], [5♦], [8♦], [5❤]
[8♣], [T♣], [6♣], [2♠], [4♠]
[9❤], [9♣], [6♦], [7♣], [A♠]
[J♣], [J❤], [9♠], [2❤], [7♠]
[7♦], [3❤], [K♦], [3♣], [8❤]
[T♦], [K♣], [5♠], [A❤], [8♠]

VI. EJEMPLOS DE POBLACIONES ANTES DE LA REPRODUCCIÓN, AL REPRODUCIRSE, AL REALIZAR EL CROSSOVER Y AL REALIZAR LA MUTACIÓN.

Crossover tiene lugar cuando dos cromosomas se reproducen, de tal manera que los dos padres fusionan produciendo dos hijos diferentes . Si dos cromosomas seleccionados se reproducen o

no estarán sujetos a la tasa de cruce elegido . La tasa de cruce será generalmente de alrededor del 70 %.

La tasa de mutación será generalmente muy pequeña , por lo general por debajo de 1 %. Una mayor tasa de mutación favorecerá exploración, sin embargo, se debe tener considerablemente pequeño para evitar la pérdida de buenos individuos.

El siguiente ejemplo muestra la población inicial y podemos observar el comportamiento en crossover con la combinación naranja y verde ,también los elementos en amarillo ejemplifican la mutación.

Población 1	Población 2
[3♦],[J♠],[5♦],[2♠],[4♠]	[3♦],[J♠],[5♦],[2♠],[4♠]
[8♣],[T♣],[6♣],[8♦],[5❤]	[8♣],[T♣],[6♣],[8♦],[5❤]
[9❤],[9♣],[9♠],[2❤],[7♠]	[9❤],[9♣],[6♦],[7♣],[A♠]
[J♣],[J❤],[6♦],[7♣],[A♠]	[J♣],[J❤],[9♠],[2❤],[7♠]
[7♦],[3❤],[5♠],[A❤],[8❤]	[7♦],[3❤],[5♠],[3♣],[8♠]
[T♦],[K♣],[K♦],[3♣],[8♠]	[T♦],[K♣],[K♦],[3♣],[8❤]

La mejor solución : Carta más alta

CONCLUSIONES

FERNANDO:

Nuevamente me entretuvo mucho realizar esta actividad de algoritmos genéticos, especialmente trabajar en el problema del agente viajero ya que realmente me puso a prueba y me costó trabajo entender cómo resolverlo. Fue gratificante ver cómo poco a poco el algoritmo va encontrando las soluciones más óptimas al problema pero sobre todo lo más interesante es ver que podemos aplicar conceptos de la "naturaleza" en programas informáticos como estos. Estoy muy sorprendido por cómo trabajan los algoritmos genéticos y en general la experiencia de investigar para resolverlo fue algo que me nutrió de mucho conocimiento

RODOLFO:

Durante esta actividad pudimos aplicar los conocimientos obtenidos en clase, los algoritmos genéticos serán de gran ayuda para el desarrollo de algunos problemas tales como son los de optimización o cuando queramos trabajar en paralelo. Ejemplo de ello lo pudimos ver en problemas NP como son el de Knapsack y agente viajero, que como vimos en algunos papers al realizar un algoritmo dinámico el proceso es mucho más lento que al usar algoritmos genéticos. Me encuentro satisfecho con este problema ya que se implementaron papers

JONATHAN:

Después de implementar los problemas aplicado un algoritmo genético se puede notar que se tiene un enfoque superior a las tradicionales de búsqueda y para resolver los problemas que se implementaron ya que con los algoritmos genéticos se tiene que encontrar la solución que es valida de forma rápida en comparación con otros enfoques, durante el desarrollo, se pudo notar como se puede usar claramente un algoritmo genético ya que muestra las combinaciones que nos permiten llegar a la solución deseada.

REFERENCIAS:

Administrator. (2016). *I-programmerinfo*. Recuperado 10 Abril, 2016, de http://www.i-programmer.info/programming/artificial-intelligence/2115-the-genetic-algorithm.html?start=1

Eprintsucmes. (2016). *Eprintsucmes*. Recuperado 10 Abril, 2016, de http://eprints.ucm.es/31285/1/AlgGenNLHE.pdf

Metuedutr. (2016). *Metuedutr*. Recuperado 16 Abril, 2016,de http://www.ceng.metu.edu.tr/~ucoluk/research/publications/tsp.pdf

Micsymposiumorg. (2016). *Micsymposiumorg*. Recuperado 10 Abril, 2016, de http://www.micsymposium.org/mics_2004/Hristake.pdf

Obitkocom. (2016). *Obitkocom*. Recuperado 10 Abril, 2016, de http://www.obitko.com/tutorials/genetic-algorithms/tsp-example.php

Plopezpozuelocom. (2016). *Plopezpozuelocom*. Recuperado 10 Abril,2016, de http://plopezpozuelo.com/pdf/msc-diss.pdf