DISTRIBUCIÓN DE REPARACIONES MEDIANTE ALGORITMOS GENÉTICOS

Philip Arias Ares, TEC
Valeria Garro Abarca, TEC
Luis Pablo Monge Angulo, TEC
Oscar Josué Ulate Alpízar, TEC
Josué Vargas Hernández, TEC

RESUMEN: *En el presente documento se explica el problema planteado para el proyecto, de forma general el problema trata de resolver una distribución de recursos mediante el uso de algoritmos genéticos.*

El uso de estos últimos y un poco de su teoría se detallarán más adelante en el artículo.

PALABRAS CLAVE: Algoritmo Genético (AG), Optimización,

1 INTRODUCCIÓN

Los algoritmos genéticos son métodos adaptativos que pueden utilizarse para resolver métodos de búsqueda y optimización. Estos están basados en el proceso genético de organismos vivos.

El principio general de estos algoritmos es que los organismos vivos evolucionan con cada generación donde los mejores genes tienen más posibilidad de pasar a la siguiente generación de individuos, lo que va mejorando la especie con el tiempo.

Los algoritmos genéticos aprovechan este principio para, a modo de evolución, encontrar soluciones óptimas para problemas matemáticos que simulan la realidad.

La manera por la que estos algoritmos funcionan, es que cada individuo es evaluado por un indicador que califica la aptitud del mismo para solucionar el problema para el que fue creado. Este indicador es llamado *fitness* y puede ser calculado de diversas maneras dependiendo del algoritmo implementado. En este proyecto fueron implementados tres algoritmos distintos para evaluar a los individuos: al azar, rueda de la fortuna y torneo.

El problema que se desea solucionar consiste en asignar la mejor distribución de agentes de reparación de aparatos eléctricos. La idea de resolver este problema es no sobrecargar de trabajo a alguno de los agentes, y al mismo tiempo generar comisiones similares para todos los agentes.

El siguiente documento explica cómo fue solucionado el problema mencionado anteriormente con los tres tipos de algoritmos implementados. Expone los objetivos generales y específicos, en detalle cada algoritmo y por último un análisis comparativo entre los resultados de estos.

2 OBJETIVOS

2.1 OBJETIVO GENERAL

Implementar una solución para asignar las rutas de los clientes a los agentes de GE de forma equitativa, implementando algoritmos genéticos.

2.2 OBJETIVOS ESPECÍFICOS

1. Implementar algoritmos genéticos que optimicen la organización de los agentes.
2. Mostrar los resultados en una interfaz gráfica amigable con el usuario final.
3. Comparar los diferentes algoritmos para encontrar el que proporciona soluciones más óptimas.

3 CONTEXTO DEL PROBLEMA

En una búsqueda para asignar recursos a diferentes solicitudes se implementará un algoritmo genético. En el problema a resolver, se trata de asignar los mejores agentes a diferentes órdenes. Cada agente puede atender diferentes tipos de servicios y cada orden requiere un servicio.

Hay que buscar la mejor manera de asignar agentes a resolver los servicios en cada orden. Aclarando que los servicios tienen una comisión y tienen una duración, por lo que no todos los agentes resuelven cualquier problema y que además no todos estarán disponibles en cualquier momento.

4 MARCO TEÓRICO

Los Algoritmos genéticos usan una analogía directa con el comportamiento natural. Trabajan con una población de individuos, cada uno representa una solución factible a un problema dado. En la naturaleza

esto equivaldría al grado de efectividad de un organismo para competir por unos determinados recursos. Cuanto mayor la adaptación de un individuo al problema, mayor será la probabilidad de que el mismo sea seleccionado para reproducirse cruzando su material genético con otro individuo seleccionado de igual forma.

Este cruce producirá nuevos individuos descendientes de los anteriores, que comparten algunas de las características de sus padres.

De esta manera se produce una nueva población de posibles soluciones, la cual reemplaza a la anterior y verifica la propiedad de que contiene una mayor proporción de mejores características en comparación con la población anterior a lo largo de las generaciones. Si el algoritmo fue bien diseñado, la población se dirigirá a una solución óptima del problema [2][4].

4.1 POBLACIÓN

En [1], se realizó un estudio teórico que arrojó que el tamaño de la población debe crecer exponencialmente con el tamaño de la ristra. Esto es, mientras más características o genes se le asigne a un individuo para realizar la evolución, más grande debe ser población. Si esta posee pocos genes que se deban modificar, se puede permitir una población inicial más pequeña, si por el contrario se tiene un tamaño de ristra grande, se debe tener una gran cantidad de individuos para que el algoritmo converja [3].

4.2 FUNCIÓN OBJETIVO

Esta es más comúnmente conocida como la función de adaptación. Esta es la función sobre la cual se juzgan los genes de cada individuo para determinar qué tan apto es el mismo para llegar a la siguiente generación. El tener una buena función objetivo no es suficiente para asegurar que un individuo pase sus genes a la siguiente generación, pero aumenta la probabilidad de que esto suceda.

Cuando esta función converge y entre las generaciones ya hay muy pocas diferencias en esta función, se considera que ya se llegó a una generación óptima para resolver el problema [3].

4.3 SELECCIÓN

Esta es una función proporcional a la función objetivo. La pasada función aumenta las probabilidades de que un individuo pase los genes a la siguiente generación, pero depende del algoritmo de selección que se está implementando.

Cada algoritmo de selección toma los datos de la función objetivo y los interpreta de una manera distinta para crear la siguiente generación de individuos [3].

4.4 MUTACIÓN

La mutación es un componente de aleatoriedad entre generaciones. Siempre se desea que los algoritmos de selección sea los que determinen la siguiente generación, pero estas mutaciones aleatorias van aumentando su importancia cuando el algoritmo genético ya va convergiendo. Esto porque los individuos dominantes van acaparando los resultados finales, por lo que aleatoriedades producen cambios importantes [3].

5 DISEÑO DEL ALGORITMO GENÉTICO

5.1 REPRESENTACIÓN

Se representó el problema mediante las siguientes estructuras de datos.

- **Población**: Una lista de diccionarios de llave y valor enteras. Cada diccionario representa un cromosoma de la población. Cada entrada del diccionario es un gen del cromosoma; La llave es el identificador de una orden y el valor es el identificador del agente. Esta estructura de datos provee un tiempo de acceso directo (en el peor caso logn) y consume menos espacio que tener un arreglo booleano de órdenes en una dimensión y agentes en otra. Es más eficiente en ese sentido ya que casi todo el arreglo se llenaría de 0's (el agente realiza una porción pequeña de órdenes), y al utilizar un diccionario se puede guardar únicamente la relación de cada usuario con cada orden.
- **Agentes Por Servicio**: Un diccionario en el que cada entrada está compuesta por una llave de tipo string y su valor una lista de enteros. La llave representa el código del servicio y el valor representa la lista de agentes que pueden atender ese servicio. Debido a que en nuestra población se almacena únicamente las relaciones de órdenes con el agente correspondiente, se necesitaba una manera de acceder eficientemente cuales agentes pueden realizar una orden. Esta estructura es constante, tiene acceso directo (en el peor caso logn) para los servicios y luego O(k) para el agente dentro de la lista de servicios. En nuestro caso no va a afectar el acceso en O(k) debido a que no se van a realizar búsquedas dentro del arreglo, van a accederse directamente (mediante un índice aleatorio) los diferentes elementos dentro de la lista. Por lo tanto, también tendremos acceso directo en la lista. La única consideración es el espacio, por cada servicio se tiene una lista con todos los agentes. Aunque esto seguro no ocurra en la realidad, se podría tener que todos los agentes están en todas las listas de los servicios; N servicios y M agentes implicaría un espacio de O(N*M).
- **Fitness**: Para representar el fitness utilizamos una lista de números flotantes. La función de evaluación será detallada en el punto 3.

5.2 GENERACIÓN INICIAL Y POSTERIORES

La población inicial se genera aleatoriamente. El algoritmo recibe como entrada el tamaño de la población, la cual se ajusta para poder almacenar un elemento adicional (cuyo motivo se explicará a continuación). La función que inicializa la población crea N (tamaño de la población) diccionarios y los llena con todos los identificadores de la órdenes asociados a un

agente aleatorio de la lista de agentes asociados al servicio (que esté asociado a esa orden).

El valor del tamaño de la población es un número impar debido a que se quiere tener ese espacio adicional para almacenar el mejor cromosoma de la generación anterior. Esto significa que en cada generación se generan N-1 sucesores directos de los cromosomas y 1 sucesor que es una copia idéntica del mejor cromosoma de la generación anterior. La copia idéntica se da para cumplir la propiedad de elitismo y no perder la mejor configuración del gen hasta el momento. Se seleccionan 2 cromosomas padres y se generan 2 sucesores de dichos padres hasta que se llena la nueva población (sin contar el mejor cromosoma de la generación actual). Las políticas de selección implementadas son las siguientes;

- **Al Azar**: Esta es la política más simple. Únicamente se generan 2 números aleatorios distintos (de 0 al tamaño de la población) que vienen a ser índices de los cromosomas dentro de la generación.
- **Rueda de la Fortuna**: Toma una cantidad de giros como parámetro. Y cada padre se genera de la siguiente manera. Se divide la población en proporción a su fitness (con respecto a la suma total). Y se simula el giro de la rueda K veces. Se toma la posición que termina ese último giro como punto para determinar cuántas veces se cae en el resto de los elementos. Entonces se va sumando por intervalo a la posición de inicio hasta recorrer toda la rueda. Cada vez que se suma por el intervalo, se determina a cuál índice corresponde esa porción de la rueda y se suma a un contador. Al final se agarran el o los elementos que tienen la mínima cantidad en el contador, y si son más de uno, se hace un torneo con esos elementos.

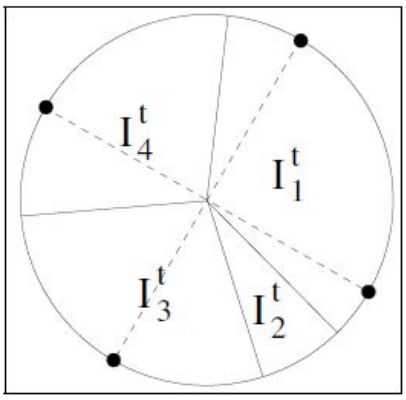

Figura 1. Modelo de la rueda de la fortuna [2].

- **Torneo**: Consiste en competencias entre dos cromosomas. Aquí se genera un número aleatorio entre 0 y la suma de los fitness de ambos competidores. Si el valor generado es de 0 al valor del fitness del primer cromosoma, entonces el otro cromosoma avanza. En vez, sí el valor generado es mayor que el fitness del primer cromosoma hasta la suma de ambos fitness. Entonces se elige el primer cromosoma. Este se hace con la intención de velar las proporciones que tienen ambos. Se quiere que el que tenga un fitness más bajo sea el que tenga más posibilidad de avanzar. Por lo tanto, se elige la contraparte si el valor generado esta dentro de la proporción del otro cromosoma. Estas competencias se realizan hasta que solo haya un cromosoma restante, el ganador.

5.3 EVALUACIÓN DE CADA GENERACIÓN

La función de evaluación o fitness se genera de la siguiente manera. Se necesita contabilizar la comisión y horas de atención por agentes. Para almacenar esta información se utiliza un diccionario con llave entera y valor de un par de enteros. La llave corresponde al identificador del agente y el par corresponde a la comisión y las horas. Se hace un acumulado de las comisiones y luego se divide por la cantidad de agentes para determinar la comisión promedio que todos los agentes deberían alcanzar. Además, para los agentes que exceden las 40 horas, se encuentra la cantidad de horas extra (*overtime*) de atención (Restando 40 a la suma de horas de atención del agente). Este total contribuye a la penalización. Con la media de comisiones se puede generar la varianza mediante la fórmula 1. Una varianza de 0 indica que todos los valores dentro del conjunto son iguales. Por lo tanto, se quiere minimizar esa varianza. Finalmente el valor fitness del cromosoma i, se da por la suma de la variancia y el producto de la penalización (*overtime,* ***OT***) con una proporción (**P**) o *bias*. Fórmula 2. Esto significa que mientras más alto sea el fitness, menos apropiada es la solución. Y por lo tanto se desea minimizar ese fitness.

$$\sigma_X^2 = (\sum_{i=1}^{n} p_i \cdot (x_i - \mu)^2) \quad (1)$$

$$F(i) = \sigma^2 + (OT * P) \quad (2)$$

5.4 CRUCES

Una vez seleccionados los padres se inicia el proceso de cruce. Este tiene sus parámetros además de los cromosomas padres; Una probabilidad de cruce y una cantidad de cruces. La probabilidad de cruce suele estar entre 0.5 y 1, mientras que la cantidad de cruces varia.

La función del cruce primero se basa en esa probabilidad de cruce para determinar si se cruzan o no los cromosomas. Si se va a realizar el cruce, se calcula un intervalo de cruces que viene a ser el tamaño del cromosoma (cantidad de genes) dividido por la cantidad de cruces más uno. Se suma uno más a la cantidad de cruces debido a que se quiere determinar en qué puntos son los que se tiene que cruzar. Por lo tanto, si la cantidad de cruces es igual a 1, se divide el tamaño del cromosoma entre 2 para determinar el punto medio del cruce.

Una vez determinado los intervalos de intersección, se generan los sucesores agarrando los genes de un

padre y copiandolos a un sucesor y los genes del otro padre y copiandolos al otro sucesor, hasta llegar a un punto de intersección. Cuando se llega a ese punto se invierte de cuál padre los sucesores reciben los genes y se repite hasta llegar al final del cromosoma.

5.5 MUTACIÓN

Por último, la fase de mutación involucra en hacer ligeras modificaciones en los dos cromosomas sucesores generados en la etapa anterior. Esta fase tiene un parámetro de probabilidad de mutación que ronda entre 0.005 y 0.02. Entonces para cada gen dentro del cromosoma se genera un número aleatorio para determinar si se muta ese gen. Esto involucra cambiar la relación de orden y agente. Es decir, cambiar cual agente está asignado a realizar esa orden. Si se determina que se va a mutar ese gen, entonces se genera otro número aleatorio para determinar el índice del nuevo agente que puede atender esa orden.

6 RESULTADOS OBTENIDOS

Figura 2. Interfaz gráfica de Distribución de Reparaciones.

Figura 3. Interfaz gráfica con los datos de los agentes y órdenes de servicio.

En las figuras anteriores se muestra el resultado de la interfaz gráfica en funcionamiento. Esta muestra cómo se despliega la información cargada de los XML.

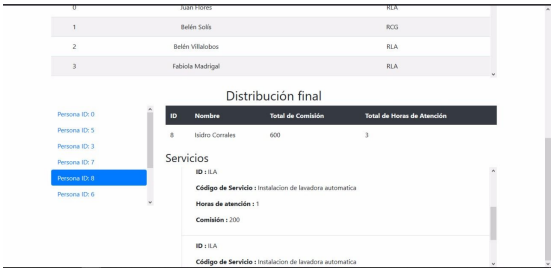

Figura 4. Resultado de la distribución de órdenes para los agentes.

En la Figura 4 se muestra la interfaz gráfica de cómo se muestra la información con el resultado del algoritmo genético sobre los valores de entrada.

7 CONCLUSIONES

1. Los algoritmos genéticos son una forma robusta de encontrar soluciones óptimas a modelos matemáticos complejos.
2. Una de las ventajas de un algoritmo genético es que con un pequeño set de reglas iniciales, se llega a soluciones robustas.
3. Es importante comprender todas las etapas de este algoritmo para poder escoger los mejores algoritmos para el modelo que se está utilizando.

8 RECOMENDACIONES

1. Utilizar algoritmos genéticos funciona para optimizar funciones matemáticas siempre y cuando se pueda tener acceso a suficientes datos.
2. Se recomienda tomar e n cuenta la cantidad de generaciones para entrenar el algoritmo. Muchas generaciones no van a retornar resultados mucho más precisos. Mientras más converge el resultado, más lento son los cambios.
3. Se recomienda tener una población inicial lo suficientemente grande como para no tener falta de variedad genética como un factor limitante a la hora de comenzar el algoritmo.
4.

9 REFERENCIAS

[1] Goldberg, J.T (1987). Genetic algorithms with sharing for multimodal function optimization. Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms and Their Applications.

[2] G, Syswerda (1991). The traveling salesman and sequence scheduling: Quality solutions using genetic edge recombination. New York. Capturado de: https://link.springer.com/article/10.1023/A:1006529012972

[3] Universidad del País Vasco. (s. f.). Algoritmos genéticos. Capturado de :

http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/temageneticos.pdf

[4] L. Davis, (1991). Handbook of Genetic Algorithms. New York. Capturado de: http://papers.cumincad.org/cgi-bin/works/paper/eaca