ALGORITMO GENÉTICO DE DISTRIBUCIÓN DE REPARACIONES PARA GENERAL ELECTRIC

INTELIGENCIA ARTIFICIAL

Elberth Adrián Garro Sánchez
e-mail: adriangarro81@gmail.com
Darío Monestel Corella
e-mail: dmcsubotai95@gmail.com
Freddy Villalobos González
e-mail: fvillalobosg94@gmail.com
Alvaro Castro Venegas
e-mail: alvarocastro74@gmail.com
Josué Jiménez Alfaro
e-mail: josuji-alfa@hotmaill.com

RESUMEN: Los algoritmos genéticos son conocidos por su variedad de uso en problemas de optimización. En este trabajo presentamos nuestra propuesta basada en la implementación de un algoritmo genético capaz de optimizar de la forma más equitativa posible la repartición de las órdenes de servicio entre los agentes de servicio tomando en cuenta una serie de restricciones. Estos algoritmos hacen evolucionar una población de individuos sometiendolas a acciones aleatorias semejantes a las que actúan en la evolución biológica bajo una serie de criterios para llegar a una solución óptima o cercana.

PALABRAS CLAVE: algoritmos genéticos, órdenes de servicio, agentes de servicio.

ABSTRACT: Genetic algorithms are known for their variety of use in optimization problems. In this paper we present our proposal based on the implementation of a genetic algorithm capable of optimizing the distribution of service orders among service agents in the most equitable way possible, taking into account a series of restrictions. These algorithms evolve a population of individuals subjecting them to random actions similar to those that act in biological evolution under a series of criteria to reach an optimal or close solutio.

KEYWORDS: genetic algorithms, service orders, service agents.

1 INTRODUCCIÓN

Los algoritmos genéticos son aquellas soluciones de software creadas y utilizadas mayoritariamente para la optimización de procesos desde un nivel personal hasta un nivel de industria.
Actualmente es una de las opciones de software predilectas en el mercado para la atención de mejora y evolución de uno o varios procesos dentro de las empresas. La teoría de la evolución desarrollada por el científico Charles Darwin es la base con la que se creó este tipo de algoritmos, los cuales constan de varias partes en las que se puede dividir la funcionalidad total de este.
Comenzando por la representación de los individuos, los algoritmos genéticos (AGs) permiten establecer las normas o parámetros que se necesitan para su clasificación, selección y posible reproducción. La parte de la población radica en la agrupación de varios individuos con características distintas entre ellos con el fin de simular la diversidad del ser humano para la supervivencia. Una parte indispensable de un AG es la función fitness o de clasificación, donde se evalúan las características propias de cada uno de los individuos con base en las condiciones ideales a las que deberían llegar y se le asigna a cada uno una calificación que cuantifique que tan cerca está de ser el más apto para la supervivencia. Una vez calificados se pasa al proceso de selección y reproducción; en esta última parte se involucran dos factores, el cruce de los genes y la mutación (está es sujeta a una probabilidad que se define previamente) y así sucede n cantidad de veces hasta hallar la solución más óptima o la óptima total.

Nuestra propuesta de software pretende resolver de una manera eficiente y eficaz la distribución de las órdenes de servicio por medio de la implementación de un AG.
Este documento contiene múltiples secciones que explican de forma detallada todo el proceso que va desde los objetivos a cumplir por el equipo de trabajo, la definición del contexto del problema, una introducción a la teoría de los algoritmos genéticos, la descripción del algoritmo creado. Por último se encuentran los resultados obtenidos por medio de la simulación, las conclusiones y recomendaciones halladas.

2 OBJETIVOS

2.1 OBJETIVO GENERAL

Desarrollar un algoritmo genético capaz de realizar una distribución de órdenes de servicio para GE.

2.2 OBJETIVOS ESPECÍFICOS

- Establecer una propuesta para la atención de los agentes de servicio de GE con respecto a las órdenes de servicio para que sean repartidas de forma equitativa.
- Implementar el AG propuesto para la optimización del tiempo de atención de las órdenes.
- Realizar una simulación gráfica en tiempo real sobre la respuesta que brinda el AG.

.

3 CONTEXTO DEL PROBLEMA

La distribución de reparaciones por parte de la empresa General Electric de Centroamérica la cual cuenta con un grupo de 2000 colaboradores que se encargan de realizar las visitas y reparaciones a domicilio alrededor de todo Costa Rica.

Las personas que estén interesadas en un servicio deben llamar al centro de servicio de GE y reservar para coordinar una visita. Todos los lunes, los agentes de servicio deben presentarse al centro de servicios de GE para que se les asigne las órdenes de servicio que deben atender durante toda la semana.

El problema es que para los coordinadores esta labor es bastante compleja ya que deben buscar repartir las órdenes lo más equitativamente posible considerando que todos ganen una comisión total por semana similar. Además, el tiempo asignado por semana no debe sobrepasar las 40 horas y no todos los agentes pueden atender todos los tipos de servicio habilitados.

Por esta razón se implementó la creación de un software para resolver este problema de distribución de servicios.

4 TEORÍA DE ALGORITMOS GENÉTICOS

Un Algoritmo Genético (AG) es un método de búsqueda que se adapta y organiza por sí mismo con una capacidad de aprendizaje proveniente de la simulación de la evolución natural [1] propuesta por Charles Darwin. Con la ley de supervivencia del más apto este algoritmo se encarga de generar nuevas poblaciones a partir de dos o más individuos.
Un AG es ideal en el procesamiento, clasificación y control de grandes y variados sets de datos [2]. Los AGs están compuestos por varias etapas desde la población, la función de fitness, el proceso de selección, cruce y mutación hasta obtener a la población final que contenga la solución más óptima o la óptima absoluta.
El rendimiento de los AGs puede verse afectado proporcionalmente a la implementación de la función de cruce. Es normal encontrar tres tipos de cruce en la mayoría de AGs como el cruce en un solo punto, el cruce de dos puntos o el cruce aleatorio. No obstante, se puede encontrar que los resultados son óptimos a pesar del uso de uno de los cruces anteriores y sin afectar el rendimiento del algoritmo.

Población: La población de los algoritmos genéticos se compone de individuos representados como vectores binarios de longitud fija [3]. Los individuos también pueden ser representados de otras formas tales como tuplas, listas u objetos, dependiendo su representación del objetivo a resolverse y es generacional. Es decir, su población se va actualizando según sus resultados de supervivencia al entorno.

Función Fitness: Los AGs utilizan la función fitness para cuantificar la adaptación de los individuos de la población, dejando a los que son más débiles en la destrucción o con pocas probabilidades de continuidad. Esta función es diseñada con base en la función objetivo.

Selección: El operador de selección está diseñado para simular el proceso de evolución de la supervivencia de los individuos con mejor índice. Es común que sólo aquellos individuos con mejor fitness sobrevivan, sin embargo, eso dejaría de lado parte de la teoría real de la evolución donde la naturaleza elige individuos con características débiles pero con una característica que combinada a un individuo fuerte evoluciona a una mejor población. De forma que es necesario hacer que este operador se ejecute con probabilidades y tome tanto individuos fuertes como débiles.

Cruce: La operación de cruce es la operación principal en un AG. El par de cromosomas intercambian sus genes de forma aleatoria en la mayoría de los casos, finalizando con una diferencia entre padres e hijos que permite igualmente en la mayoría de ocasiones llegar a la población con la solución óptima.

Mutación: Esta operación hace referencia a la generación de un nuevo individuo mediante el cambio de uno o varios de sus genes de una forma aleatoria de sus posiciones. Está diseñado con el fin de mantener la diversidad de la población.
A pesar de que el cruce y la mutación son los principales operadores genéticos, es posible usar otros como reagrupación, colonización-extinción o migración.

Condición de parada: proceso se repite por generaciones hasta que se encuentre la solución más óptima o la óptima total. Las condiciones comunes de parada incluyen:
– La solución encontrada satisface el objetivo.
– Tamaño máximo de generaciones alcanzado.
– Presupuesto asignado (tiempo/dinero) alcanzado.
– El fitness de la solución de mayor rango está llegando o ha alcanzado un nivel tal que las iteraciones sucesivas ya no producen mejores resultados.

5 DISEÑO DEL ALGORITMO GENÉTICO

5.1 REPRESENTACIÓN

El algoritmo genético tiene genes los cuales serán representados mediante un vector de seis entradas, donde cada entrada representa un servicio de los que se encuentran en oferta. Entonces el valor de cada entrada representa la cantidad de servicios de cada tipo que va a tener el gen, en otras palabras estos genes representan los agentes de la empresa. De esta manera se indica cuantos servicios va a realizar el agente relacionado a su tipo de servicio que ofrece.

Luego se procede a calcular el costo del gen o del agente de servicio, tomando en cuenta de que el tiempo no sobrepase las cuarenta horas.

5.2 GENERACIÓN INICIAL

Se procede a crear una población inicial de genes cuya cantidad de horas no pase de cuarenta esto se realiza de manera aleatoria para generar la cantidad de agentes y de órdenes.

5.3 EVALUACIÓN DE CADA GENERACIÓN

Estos genes creados estarán en archivo de formato JSON, donde luego se convertirá en un arreglo que tiene las llaves de cada gen y se le realiza un ordenamiento aleatorio para irlos iterando en parejas de acuerdo a su índice.

5.4 CRUCES

A las parejas definidas en el punto anterior se procede a realizarles un cruce, donde toma de manera aleatoria algunas entradas del primer gen y del segundo gen para mezclarlas y formar un nuevo gen.

De este modo se cuenta con 3 genes: la pareja inicial y el gen resultante del cruce, luego se calcula cual tiene un mejor fitness, en caso de que el gen hijo tenga un mejor fitness que sus padres este de manera aleatoria se sustituye por algunos de sus predecesores y así sucesivamente con cada una de las parejas iteradas.

En nuestro caso el fitness se calcula con un valor heurístico, representado como la suma de todos los costos de las órdenes y esto se divide entre la cantidad de agentes, es decir se realiza la distribución de los costos. Por ejemplo tenemos seis servicios y tenemos a dos empleados de GE, donde cada servicio tiene un costo de mil colones, lo cual sumaría seis mil colones entre los servicios disponibles, entonces la forma más justa es que cada empleado se gane tres mil colones, ese valor sería el heurístico. De esta manera al calcular el fitness se tomaría el costo del gen y dividirlo al valor heurístico asociado en valor absoluto, de esta manera nos indicaría que tan bien repartido están los costos.

Si el gen tiene un valor muy similar al valor heurístico, esto indica que la repartición es justa, esto se da cuando la división del costo y el valor heurístico en valor absoluto da un valor cercano a cero lo cual indica que tenemos un buen fitness.

$$H = \sum_{i}^{101} 0_i[cost] / \ |A|$$

$$F(g) = |g[cost] - H|$$

$$F(g) \rightarrow 0 \ _{is \ \ better}$$

5.5 MUTACIÓN

En este caso se realiza una mutación con el 1% de probabilidad para alterar o cambiar alguno de los valores de entrada de algún gen de manera aleatoria.

6 RESULTADOS OBTENIDOS

Luego se toma la mitad de la población y se repite el proceso de hacer los cruces, realizar la mutación y nuevamente se toma la mitad de la población tomando siempre a los genes con mejor fitness y así sucesivamente hasta tener llegar a un 25% de la población original, donde se pretende que tenga los genes con costos distribuidos de manera equitativa.

Para cada agente agente debe haber genes válidos, por ejemplo un agente puede atender el servicio 2 pero no el servicio 5, por lo que se debe validar si son compatibles, también está el caso de que un agente sea compatible con varios genes para ello a este grupo selecto se le realiza una especie de balanceo para repartir las órdenes entre los agentes de manera equitativa y que sean compatibles para ajustarse a la demanda establecida siempre y cuando de un valor muy cercano a cero.

6.1 EJECUCIÓN DE LA APLICACIÓN

Ejecutamos el Index.html donde se encuentra la aplicación y nos muestra la siguiente ventana.

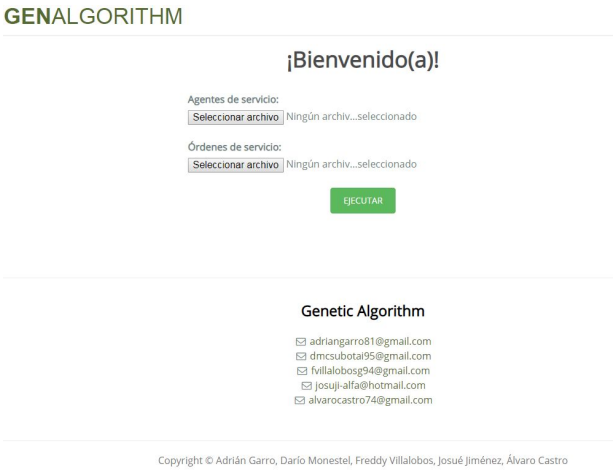

figura 1 - Aplicación GenAlgorithm

Luego ya sea seleccionando archivos en formato JSON para los agentes y órdenes definidos o mediante comandos de voz cargamos los parámetros requeridos. Por ejemplo mediante comandos de voz le decimos a la aplicación: “Agentes 50”, “Crear agentes”, “Órdenes 50” “Crear órdenes” (para cada comando de voz se debe presionar la tecla 1 para habilitar el micrófono en el navegador Chrome y luego la tecla 0 para finalizar la ejecución de cada comando de voz), luego seleccionamos el botón “Ejecutar” para generar las soluciones.

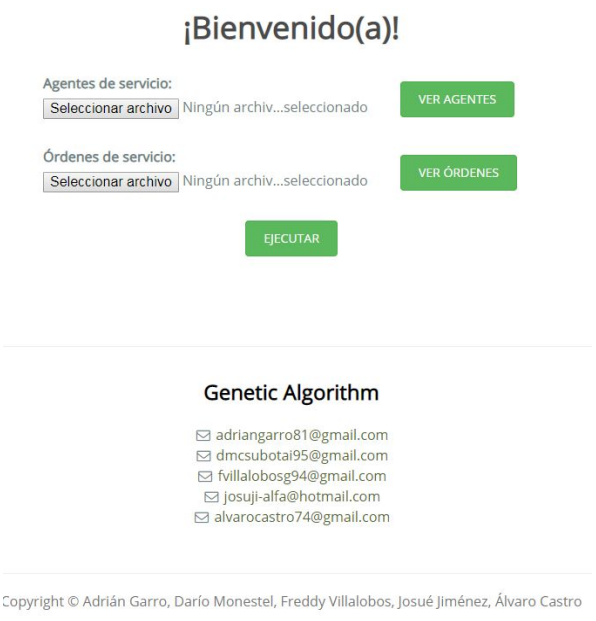

figura 2 - Aplicación GenAlgorithm

figura 3 - Aplicación GenAlgorithm

En la figura anterior se nos muestra la forma en que se atendió la demanda de las órdenes de servicio.
Los botones verdes que se muestran a la derecha nos muestran los nombres de los clientes y el servicio solicitado por los mismos.

figura 4 - Aplicación GenAlgorithm

Toda lista de soluciones y de las órdenes de agente cuentan con un buscador para facilitar la localización de algún cliente o agente de servicio si se cuenta con una gran demanda.

7 CONCLUSIONES

Los resultados revelados por la implementación de nuestra propuesta arrojan una serie de conclusiones como las siguientes:

- Los algoritmos genéticos son una alternativa bastante buena para resolver problemas que de otra manera serían muy complejos con un enfoque tradicional en problemas que se pueden aplicar perfectamente no solo en ciencias de la computación como en casos cotidianos.

- Es un método genérico que puede permitir adaptarse a problemas NP es un buen intento o acercamiento al encontrar una solución óptima.

- Los algoritmos genéticos tienen como ventaja, que no incorporan ninguna restricción a la función objetivo.

- En el desarrollo de un algoritmo genético se realiza un análisis, se plantea la estructura del cromosoma, la población con la que se trabajara, los métodos de evaluación, selección, cruzamiento y mutación.

- El problema de General Electric es solo una de todas las aplicaciones de estas tecnologías en problemas que de otra manera son imposibles de resolver para las empresas.

8 RECOMENDACIONES

- Se recomienda disminuir la población inicial del algoritmo para converger a la solución esperada de acuerdo a cada iteración.

- Para el desarrollo de un AG es necesario realizar un estudio del tamaño de la población y la cantidad de evoluciones que se deben utilizar en el modelo a implementar, debido a que esto influye en el comportamiento del AG.

- Para realizar evaluaciones entre evaluaciones y resultados de los AG, es mejor trabajar con poblaciones iniciales que sean generadas mediante resultados historicos, de esta manera se garantiza que el punto de partida del análisis para las diferentes poblaciones será homogéneo.

- Se recomienda el uso de ED como los árboles y el uso de archivos de formato JSON para el uso de un lenguaje de programación como javaScript.

- JavaScript facilita mucho la creación y la manipulación de la interfaz gráfica y comandos de voz.Ayuda además a abstraer el problema en alto nivel. Brinda un balance entre rendimiento y apariencia.

9 REFERENCIAS

1] Y. Zhu, “Hybrid electric car fuel consumption optimization research based on improved genetic algorithm,” 2017, pp. 509–512, cited By 0. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=

2-s2.0-85032830871&doi=10.1109%2fICITBS.2016.94&partnerID=40&md5=a5ed76e74d6895bfa80b12af5b598c0f

[2] J. Stender, “Introduction to genetic algorithms,” in IEE Colloquium on Applications of Genetic Algorithms, 1994, pp. 1/1–1/4.

[3] F. Li, Q.-H. Liu, F. Min, and G.-W. Yang, “A new crossover operator based on the rough set theory for genetic algorithms,” in 2005 International Conference on Machine Learning and Cybernetics, vol. 5, Aug 2005, pp. 2907–2912 Vol. 5.

[4] A. Shala and M. Bruqi, “Trajectory tracking of mobile robot using designed optimal controller,” International Journal of Mechanical Engineering and Technology, vol. 8, no. 8, pp. 649–658, 2017, cited By 0. [Online]. Available:

https://www.scopus.com/inward/record.uri?eid=2-s2.0-85029077561&partnerID=40&md5=43cfcc903731ecd36233d648c284b56d