****

**CONCURSO NACIONAL DE COMPUTACION**

, septiembre 2020

**Tutor:**

MSc. Andy Morfa Hernández ([amorfa@uclv.cu](mailto:amorfa@uclv.cu))

**Autores:**

Denis Rojas García ([drojas@uclv.cu](mailto:drojas@uclv.cu))

José Manuel Cassola Bacallao ([jcassola@uclv.cu](mailto:jcassola@uclv.cu))

**Carrera:**

Ingeniería Informática (5to Año)

**Tema:**

Visualización e interacción con las metaheurísticas Búsqueda Tabú y Algoritmo Genético en el Problema del Viajero Vendedor y el Problema de Ruteo de Vehículos

# Resumen.

El proceso de solución de los problemas de optimización constituye una ciencia en constante desarrollo. Las técnicas de interacción visual para apoyar algoritmos metaheurísticos representan una atractiva variante en el progreso de la misma, gracias a las ventajas que pueden ofrecer el análisis e interacción de un usuario en tiempo de ejecución de un algoritmo.

A partir del estudio realizado sobre esta rama que combina la informática y las matemáticas, se definieron importantes interacciones visuales a integrar en la metaheurísticas Algoritmo Genético y Búsqueda Tabú, con el objetivo de optimizar los resultados al resolver el Problema del Viajero Vendedor y el Problema de Ruteo de Vehículos. Dicho procedimiento trajo como resultado el desarrollo de los softwares GAVis y TSVis. Como parte del desarrollo de dichos softwares se realizó el análisis pertinente que demostró la eficiencia de la solución propuesta.

**Palabras Claves:** problemas, optimización, metaheurística, Algoritmo Genético, Búsqueda Tabú, software.

# 

# Abstract.

The process of solving optimization problems is a constantly developing science. Visual interaction techniques to support metaheuristic algorithms represent an attractive variant in its progress, thanks to the advantages that the analysis and interaction of a user at runtime of an algorithm can offer.

From the study carried out on this branch that combines computer science and mathematics, important visual interactions were defined to be integrated into the Genetic Algorithm and Tabu Search metaheuristics, with the aim of optimizing the results when solving the Problem of the Traveling Salesman and the Problem of Vehicle Routing. This procedure resulted in the development of the GAVis and TSVis software. As part of the development of said software, the pertinent analysis was carried out that demonstrated the efficiency of the proposed solution.

**Keywords:** problems, optimization, metaheuristics, Genetic Algorithm, Tabu Search, software.

# Introducción.

Desde hace varias décadas se ha venido investigando la formación de una gran variedad de problemas de optimización que requieren ser abordados con metodologías generales sin dejar de considerar explícitamente el universo de variables de decisión que cada situación involucra. Estos problemas no resultan ser tan difíciles de entender, pero sus esquemas de solución, a través de técnicas exactas, son apenas aplicables cuando el número de variables de decisión es relativamente pequeño, y en general gran parte de los problemas cotidianos no cumplen esta condición. Ante dicha realidad, muchos esfuerzos en investigación se han concentrado en desarrollar nuevas técnicas y procedimientos heurísticos, los cuales han tenido un gran auge, sin duda, debido a la necesidad de disponer de herramientas que permitan resolver problemas tales como los de optimización.(de la Cruz H et al. 2003)

Para ello, se hace uso de los métodos metaheurísticos, los cuales resuelven problemas de búsqueda y optimización encontrando una solución que, aunque no necesariamente es la óptima, es una buena solución y se obtiene en un tiempo razonable. (Alberto et al. 2016)

Problemas de estas características son los famosos Problema del Viajero Vendedor (TSP, *Travelling Salesman Problem*) (Applegate et al. 2006) y Problema de Ruteo de Vehículos (VRP, *Vehicle Routing Problem*) (Dantzig & Ramser 1959), y algunos de los algoritmos metaheurísticos más famosos y utilizados en diferentes problemáticas es el conocido Algoritmo Genético (GA, *Genetic Algorithm*) (Holland 1975) y la Búsqueda Tabú (TS, *Tabu Search*) (Glover 1986).

Sin embargo, generalmente los softwares desarrollados que utilizan el Algoritmo Genético, la Búsqueda Tabú u otras metaheurísticas para resolver estos problemas poseen escasa o ninguna interacción visual con el usuario por lo que deja al mismo con pocas opciones de modificación y/o interacción más allá de la ejecución del programa.

# Desarrollo.

## Algoritmo Genético (GA).

Los algoritmos genéticos presentan una propuesta eficaz motivada en la observación de que la evolución natural ha sido extraordinariamente exitosa en desarrollar especies complejas y bien adaptadas por medio de un mecanismo simple (Vélez & Montoya 2007). La técnica emula la evolución natural para explorar con eficiencia el espacio de búsqueda con el supuesto de que unos individuos con ciertas características son aptos para sobrevivir y transmiten estas características a su descendencia. (Youssef et al. 2001)

Los algoritmos genéticos operan sobre una población o conjunto de soluciones representadas como cadenas binarias o cromosomas. Durante la ejecución, el algoritmo cruza los individuos de mayor aptitud para renovar la población y elimina los de menor aptitud. Al final, el cromosoma de mayor aptitud es la solución al problema.

## Terminología del GA.

Según (Mitchell 1998) algunos conceptos básicos son:

* **Cromosoma***.* Cadena binaria que representa un individuo o solución, donde cada elemento en la cadena se conoce como gen (Ilustración 5a).
* **Población***.* Conjunto finito de cromosomas (Ilustración 5b).
* **Aptitud***.* Criterio que evalúa la calidad de un cromosoma. A mayor aptitud, mejor la solución y mayor la probabilidad de que sobreviva y transmita sus características a su descendencia.
* **Cruce**. Operación por medio de la cual se producen nuevos descendientes a partir de dos cromosomas padre seleccionados al azar (Ilustración 6a).
* **Mutación**. En esta operación se seleccionan al azar y se cambian uno o más genes en el cromosoma; ocurre con probabilidades muy bajas (Ilustración 6b).

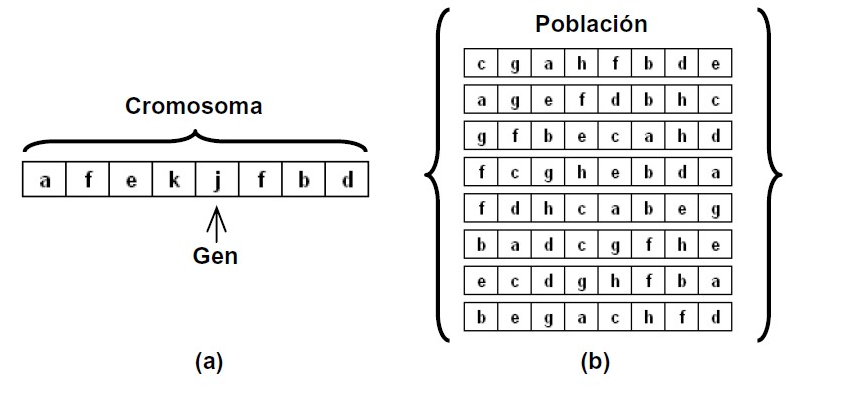


Ilustración 1: Representación de la población en un algoritmo genético (Vélez & Montoya 2007)

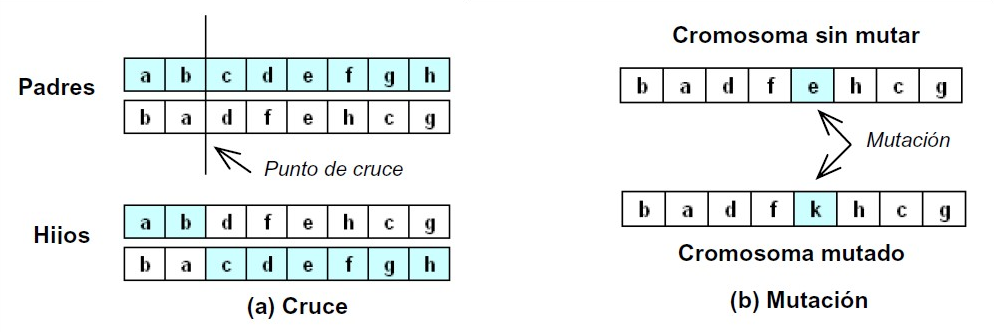


Ilustración 2: Operaciones básicas en algoritmos genéticos (Vélez & Montoya 2007).

## Algoritmo Búsqueda Tabú.

La Búsqueda Tabú (TS, *Tabu Search*) es un procedimiento heurístico de memoria adaptativa para la exploración de óptimos globales en problemas de optimización mono-objetivos. El método se basa en el principio que es mejor realizar un mal movimiento, cuando este es realizado de forma informada, que un buen movimiento ejecutado de forma aleatoria. Está siendo aplicado exitosamente en las últimas décadas en multitud de problemas complejos (continuos y discretos, lineales y no lineales convexos y no convexos, etc.) (Sicilia et al. 2016)

Búsqueda Tabú explora el espacio de soluciones a través de repetidos movimientos desde una solución a la mejor de sus vecinas tratando de evitar los óptimos locales. Para un problema mono-objetivo, TS realiza una búsqueda por entornos en la cual se desplaza en cada iteración a la mejor solución no tabú del vecindario de la solución actual. Los principales atributos de cada solución visitada son almacenados en una lista tabú por un determinado número de iteraciones para evitar que estas soluciones sean revisitadas, es decir, para evitar ciclos en la búsqueda por entornos. Así, un elemento del vecindario de la solución actual es declarado tabú (es decir, es prohibido) si alguno de sus atributos está en la lista tabú. En general, un método basado en Búsqueda Tabú requiere de los siguientes elementos (Alancay et al. 2016):

1. **Solución inicial**. La búsqueda debe comenzar desde una solución inicial que podría ser cualquier solución admisible que satisfaga las restricciones del problema. Una buena solución inicial podría acelerar la búsqueda con el consiguiente ahorro de tiempo. Dicha solución puede ser generada aleatoriamente o utilizando funciones ávidas o *greedy functions*[[1]](#footnote-1).
2. **Movimiento**. Un movimiento es un procedimiento aleatorio o determinístico por el que se genera una solución admisible a partir de la solución inicial. Usualmente, este procedimiento es sencillo para el caso de problemas combinatorios, pero mucho más complejo para el caso de problemas de optimización continuos.
3. **Vecindad o vecindario**. Dada una solución *s*, la vecindad *N(s)* es el conjunto de todas las soluciones admisibles que pueden ser generadas por la ejecución de un movimiento sobre la solución actual *s*. Este conjunto suele ser numerable para problemas combinatorios y, en aquellos casos en los que *N(s)* sea grande, se suele operar con un subconjunto de éste. Para problemas continuos, los posibles vecinos son no numerables y se debe ser más creativo para definir *N(s).*
4. **Lista tabú.** Es un mecanismo de memoria adaptativa que trata de evitar que la búsqueda entre en un ciclo o quede atrapada en un óptimo local. Una vez que un movimiento, que genera una nueva solución, es aceptado, su movimiento inverso se añade a la lista tabú y permanece en ésta un número determinado de iteraciones. Si el tamaño de la lista tabú es pequeño, entonces la búsqueda se intensifica en una determinada área del espacio, mientras que si el tamaño de la lista es grande se enfatiza la búsqueda en diferentes regiones del espacio de soluciones.

**Criterio de parada.** En general, la búsqueda termina después de un número determinado de iteraciones, después de un tiempo de computación predefinido o cuando se alcanza un número dado de iteraciones sin mejorar la mejor solución.

## Estudio Experimental Comparativo.

El objetivo fundamental que presenta la realización del estudio es poder establecer una comparación entre los resultados oficiales registrados, disponibles en TSPLIB (<http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html>) y los alcanzados por GAVis y TSVis, ejecutando los softwares desarrollados sin asistencia del usuario y con asistencia del mismo, a través de las interacciones implementadas, para valorar la efectividad y eficacia del sistema.

En la realización del siguiente estudio experimental se utilizaron diferentes ficheros correspondientes a cada problema. ([Ver Tabla 1](#_Tabla_1:_Ficheros))

El sistema se ejecutó en un ordenador con las siguientes propiedades:

* Procesador: Intel(R) Core (TM) i7-6700HQ CPU @ 2.60GHz (8CPUs).
* Memoria: 8192 MB RAM.
* Sistema Operativo: Windows 10 Pro 64 bits (10.0, compilación 18362.239)

Los parámetros necesarios para la ejecución de la Búsqueda Tabú son los siguientes:

* **Criterio de Diversificación (Diversification Criteria).** Es el componente con el que dispone el algoritmo para salir de los mínimos locales porque alcanzado éste se demorará un tiempo en salir del mismo, realiza una solución aleatoria alejada de la solución actual. El criterio se establece en segundos. Solo para ficheros del TSP.
* **Vecindario (Neighborhood).** Contiene los tres movimientos principales para la ejecución del algoritmo. Solo para problemas TSP.
  + ***SwapMove*:** Consiste en el intercambio de dos ciudades en la trayectoria.
  + ***InsertMove*:** Consiste en la inserción de una ciudad que se encuentra en una posición predeterminada a otra posición en el camino.
  + ***InvertMove*:** Consiste en invertir el orden de secuencia de las ciudades contenida en el camino.
* **Tamaño de la Lista Tabú (Size TabuList).** Esta lista contiene los movimientos que se han hecho durante la ejecución del algoritmo. Estos movimientos están marcados como tabú y el tamaño de esta lista va a determinar significativamente el rendimiento del algoritmo para que pueda salir de los mínimos locales.

Los parámetros necesarios para la ejecución del Algoritmo Genético son los siguientes:

* **Población (Population).** Es el conjunto de cromosomas (soluciones) generados después de una iteración.
* **Generaciones (Generations).** Se forman nuevas poblaciones a partir de poblaciones anteriores, ya sea seleccionando un cromosoma completo (generalmente de alta aptitud), modificando un poco un cromosoma existente (mutación) o, más comúnmente, generando nuevos cromosomas a partir de cromosomas “padres” (cruzamiento).
* **Selector Natural (Natural Selector).** Su función es garantizar que los cromosomas de mejor rendimiento (mayor aptitud) tengan una mayor probabilidad de ser utilizados para reproducir la próxima generación.
* **Cruzamiento (Crossover).** Se selecciona al azar dos cromosomas de la población y se “cruzan”, escogiendo de manera aleatoria un gen y luego intercambiando dicho gen y todos los genes subsecuentes entre los dos cromosomas. El resultado será dos descendientes que combinan las características de ambos cromosomas “padres”.
* **Mutación (Mutation).** Modifica al azar parte (genes) del cromosoma de los individuos (sin incumplir las restricciones del problema) y permite alcanzar zonas del espacio de búsqueda que no estaban cubiertas por los individuos de la población actual. Es una estrategia para escapar de los mínimos locales potenciales y de generar nuevo material de cruzamiento.
* **Proporción (Rate).** Es la probabilidad de que ocurra la funcionalidad del parámetro al que está vinculado (selección natural, cruzamiento, mutación).

### Travelling Salesman Problem.

Para la ejecución sin asistencia del usuario de este problema en el software TSVis, se tomaron 200 000 iteraciones sin pausa, con un criterio de diversificación de 1 segundo, un tamaño de la Lista Tabú de 5 y se seleccionó el Vecindario *InvertMove*, siempre partiendo de una solución inicial generada por el algoritmo *Greedy*. Además, se realizaron 10 ejecuciones con cada fichero para escoger el mejor resultado. ([Ver Tabla 2](#_Tabla_2:_Resultados_1)).

Para la ejecución con asistencia del usuario se utilizaron 150 000 iteraciones en conjunto con la variación de los parámetros del criterio de diversificación, el tamaño de la Lista Tabú y la selección del Vecindario y se podrá pausar y continuar según las decisiones del usuario. Se realizaron 10 ejecuciones con cada fichero para escoger el mejor resultado. ([Ver Tabla 2](#_Tabla_2:_Resultados_1))

Para la ejecución sin asistencia del usuario de este problema en el software GAVis se seleccionaron 5000 generaciones sin pausa, un tamaño de población calculado por el algoritmo acorde a la cantidad de ciudades en el problema, un Selector Natural de tipo *Ranking*, un Cruzamiento de tipo *Grafenstettes* y una Mutación de tipo *Segment Swapping*. La proporción para el Selector Natural y la Mutación es de un 50%. Finalmente, se contabilizó el tiempo de ejecución del algoritmo en segundos. ([Ver Tabla 4](#_Tabla_4:_Resultados))

Para la ejecución con asistencia del usuario se seleccionaron 3500 generaciones, el resto de los parámetros iniciales fueron los mismos que los de la ejecución sin asistencia del usuario. Además, fueron realizadas diversas paradas durante la ejecución del algoritmo en las cuales se modificaron los parámetros y sus respectivas proporciones, así como variaciones en la ruta obtenida a través de la inserción y/o eliminación de aristas. ([Ver Tabla 4](#_Tabla_4:_Resultados))

### Vehicle Routing Problem.

Para la ejecución sin asistencia del usuario de este problema en el software TSVis, se tomaron 200 000 iteraciones sin pausa, con tamaño de la Lista Tabú de 3, siempre partiendo de una solución inicial generada por el algoritmo Greedy. Además, se realizaron 10 ejecuciones con cada fichero para escoger el mejor resultado. ([Ver Tabla 3](#_Tabla_3:_Resultados))

Para la ejecución con asistencia del usuario se utilizaron 150 000 iteraciones en conjunto con la variación de los parámetros del tamaño de la Lista Tabú y se podrá pausar y continuar según las decisiones del usuario. Se realizaron 10 ejecuciones con cada fichero para escoger el mejor resultado. ([Ver Tabla 3](#_Tabla_3:_Resultados))

Para la ejecución sin asistencia del usuario de este problema para el software GAVis, se tomaron 5000 generaciones sin pausa, un tamaño de población de 250, un Selector Natural de tipo *Ranking*, un Cruzamiento de tipo *Default* y una Mutación de tipo *Two Way*. La proporción para el Selector Natural, el Cruzamiento y la Mutación es de un 50%. Finalmente, se contabilizó el tiempo de ejecución del algoritmo en segundos. ([Ver Tabla 5](#_Tabla_5:_Resultados))

Para la ejecución con asistencia del usuario se seleccionaron 3500 generaciones, el resto de los parámetros iniciales fueron los mismos que los de la ejecución sin asistencia del usuario. Además, fueron realizadas diversas paradas durante la ejecución del algoritmo en las cuales se modificaron los parámetros y sus respectivas proporciones. ([Ver Tabla 5](#_Tabla_5:_Resultados))

## Pruebas de caja negra realizadas a las aplicaciones.

Las pruebas de caja negra permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos. Estas pruebas se realizan mediante la técnica de partición de equivalencia que consiste en dividir el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.(Sánchez Peño 2015)

1. Caso de prueba para el Caso de Uso “Cargar Instancia” (TSP) y los resultados de la ejecución de las clases de entrada propuestas en TSVis. (Ver [Anexo 3](#_Anexo_3:_Pruebas), Tablas [6](#_Tabla_6:_Clasificación) y [7](#_Tabla_7:_Resultados))
2. Caso de prueba para el Caso de Uso “Ejecutar Búsqueda Tabú” (TSP) y los resultados de la ejecución de las clases de entrada propuestas en TSVis. (Ver Tablas [8](#_Tabla_8:_Clasificación) y [9](#_Tabla_9:_Resultados))
3. Caso de prueba para el Caso de Uso “Cargar Fichero” (TSP) y los resultados de la ejecución de las clases de entrada propuestas en GAVis. (Ver Tablas [10](#_Tabla_10:_Clasificación) y [11](#_Tabla_11:_Resultados))
4. Caso de prueba para el Caso de Uso “Ejecutar Algoritmo Genético” (TSP) y los resultados de la ejecución de las clases de entrada propuestas en GAVis. (Ver Tablas [12](#_Tabla_12:_Clasificación) y [13](#_Tabla_13:_Resultados))

# Conclusiones.

Durante todo el transcurso de esta investigación se obtuvieron como resultado dos nuevos programas de cómputo que dan solución al Problema del Viajero Vendedor y al Problema de Ruteo de Vehículos, a través del uso de las metaheurísticas Algoritmo Genético y la Búsqueda Tabú, cumpliéndose de esta forma los objetivos planteados, ya que:

1. Se presentaron los aspectos teóricos de ambos problemas, haciéndose énfasis en los problemas de optimización y su complejidad. Además, se describieron las metaheurísticas Algoritmo Genético y Búsqueda Tabú, las interacciones visualizaciones, en específica con los grafos, partiendo del análisis visual de datos.
2. Se realizó la implementación de los softwares TSVis y GAVis, que permiten la interacción entre el usuario y los algoritmos de búsqueda metaheurística en la solución del TSP y el VRP.
3. Se demostró la efectividad y eficacia de ambos softwares implementados mediante la comparación de resultados, logrando así buenos resultados con la incorporación del usuario a través de las interacciones visuales.
4. Se realizaron pruebas de caja negra a cada producto, validando así el comportamiento de ambas aplicaciones.

# Recomendaciones

Considerando los resultados derivados del proceso investigativo y el desarrollo de los sistemas GAVis y TSVis se recomienda:

1. Implementar nuevas interacciones para el Problema de Ruteo de Vehículos para poder así restringir el espacio de búsqueda y obtener mejores soluciones.
2. Implementar el uso del parámetro de Criterio de Diversificación en el Problema de Ruteo de Vehículos en el software TSVis para evitar los óptimos locales.
3. Indagar sobre nuevos algoritmos de búsquedas de mayor complejidad que propicien soluciones iniciales más eficientes que el algoritmo Greedy.
4. Emplear de manera docente GAVis y TSVis en el aprendizaje de los estudiantes, principalmente en temáticas como la búsqueda heurística y metaheurística en la asignatura Inteligencia Artificial.
5. Profundizar en el estudio de nuevas técnicas de visualización para heurísticas reconocidas que han demostrado buenos resultados resolviendo cada uno de estos problemas.

# Referencias Bibliográficas.

Alancay, N., Villagra, S. & Villafra, A., 2016. Algoritmos metaheurísticos trayectoriales para optimizar problemas combinatorios. , pp.56–75.

Alberto, J. et al., 2016. Métodos de búsqueda usando los algoritmos de enjambre de partículas y genético. , pp.52–60.

Applegate, D.L. et al., 2006. The Travelling Salesman Problem.

Dantzig, G.B. & Ramser, J.H., 1959. The Truck Dispatching Problem.

Glover, F., 1986. Future paths for integer programming and links to artificial intelligence. *Computer and Opetations Research*, pp.5:533–549.

Holland, J., 1975. Adaptation in Natural and Artificial Systems. *University of Michigan Press*.

De la Cruz H, J.J. et al., 2003. Análisis comparativo de las aproximaciones heurísticas Ant-Q, recocido simulado y búsqueda tabú en la solución del problema del agente viajero. *Ingeniería y desarrollo: revista de la División de Ingeniería de la Universidad del Norte*, pp.141–157.

Mitchell, M., 1998. *L. D. Davis, Handbook of Genetic Algorithms*, Elsevier Science B.V.

Sánchez Peño, J.M., 2015. *Pruebas de Software. Fundamentos y Técnicas.* Universidad Politécnica de Madrid.

Sicilia, J.A. et al., 2016. An optimization algorithm for solving the rich vehicle routing problem based on Variable Neighborhood Search and Tabu Search metaheuristics. *Journal of Computational and Applied Mathematics*, 291, pp.468–477.

Vélez, M.C. & Montoya, J.A., 2007. METAHEURÍSTICOS: UNA ALTERNATIVA PARA LA SOLUCIÓN DE PROBLEMAS COMBINATORIOS EN ADMINISTRACIÓN DE OPERACIONES. *Revista EIA.Esc.Ing.Antioq no.8*.

Youssef, H., Sait, S.M. & Adiche, H., 2001. Evolutionary algorithms, simulated annealing and tabu search: a comparative study. , 14, pp.167–181.

# Anexos.

## Anexo I: Ficheros utilizados por las aplicaciones TSVis y GAVis.

##### Tabla 1: Ficheros TSP y VRP

|  |  |
| --- | --- |
| **TSP (.tsp)** | **VRP (.vrp)** |
| eil51 | B-n34-k5 |
| berlin52 | P-n45-k5 |
| st70 | A-n53-k7 |
| eil76 | A-n69-k9 |
| kroA100 | B-n78-k10 |
| ch150 | E-n101-k8 |
| a280 | M-n121-k7 |

## 

## Anexo II: Resultados experimentales de las aplicaciones TSVis y GAVis

##### Tabla 2: Resultados del estudio experimental en ficheros TSP utilizando TSVis

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Fichero (.tsp)** | **Ciudades** | **Soluciones** | | | | **Tiempo de Ejecución sin asistencia (s)** |
| **TSPLIB** | | **TSVis (sin asistencia)** | **TSVis (con asistencia)** |
| eil51 | 51 | 426.0 | 429.0 | | 426.0 | 50.71 |
| berlin52 | 52 | 7542.0 | 7734.0 | | 7542.0 | 51.451 |
| st70 | 70 | 675.0 | 680.0 | | 675.0 | 93.095 |
| eil76 | 76 | 538.0 | 549.0 | | 541.0 | 116.683 |
| kroA100 | 100 | 21282.0 | 21320.0 | | 21282.0 | 250.637 |
| ch150 | 150 | 6528.0 | 6727.0 | | 6579.0 | 779.761 |
| a280 | 280 | 2579.0 | 3877.0 | | 2654.0 | 4012.684 |

##### 

##### Tabla 3: Resultados del estudio experimental en ficheros VRP utilizando TSVis

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Fichero (.vrp)** | **Ciudades** | **Vehículos** | **Soluciones** | | | **Tiempo de Ejecución sin asistencia (s)** |
| **TSPLIB** | **TSVis (sin asistencia)** | **TSVis (con asistencia)** |
| P-n45-k5 | 45 | 5 | 510 | 629.49 | 523.51 | 15.516 |
| A-n53-k7 | 53 | 7 | 1010 | 1232.93 | 1080.15 | 35.581 |
| A-n69-k9 | 69 | 9 | 1159 | 1288.19 | 1194.88 | 69.865 |
| B-n78-k10 | 78 | 10 | 1221 | 1425.73 | 1399.21 | 139.077 |
| E-n101-k8 | 101 | 8 | 815 | 932.94 | 887.00 | 146.614 |
| M-n121-k7 | 121 | 7 | 1034 | 1214.24 | 1164.81 | 299.684 |

##### Tabla 4: Resultados del estudio experimental en ficheros TSP utilizando GAVis

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Fichero (.tsp)** | **Ciudades** | **Soluciones** | | | | **Tiempo de Ejecución sin asistencia (s)** |
| **TSPLIB** | | **GAVis (sin asistencia)** | **GAVis (con asistencia)** |
| eil51 | 51 | 426.0 | 437.0 | | 426.0 | 104.668 |
| berlin52 | 52 | 7542.0 | 7722.0 | | 7542.0 | 152.609 |
| st70 | 70 | 675.0 | 737.0 | | 682.0 | 457.739 |
| eil76 | 76 | 538.0 | 571.0 | | 547.0 | 538.448 |
| kroA100 | 100 | 21282.0 | 24964.0 | | 21388.0 | 992.416 |
| ch150 | 150 | 6528.0 | 7194.0 | | 6567.0 | 2881.334 |
| a280 | 280 | 2579.0 | 2869.0 | | 2586.0 | 3594.115 |

##### Tabla 5: Resultados del estudio experimental en ficheros VRP utilizando GAVis

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Fichero (.vrp)** | **Ciudades** | **Vehículos** | **Soluciones** | | | **Tiempo de Ejecución sin asistencia (s)** |
| **TSPLIB** | **TSVis (sin asistencia)** | **TSVis (con asistencia)** |
| B-n34-k5 | 34 | 5 | 788.0 | 804,33 | 796,17 | 294.886 |
| A-n53-k7 | 53 | 7 | 1010.0 | 1217,20 | 1157,86 | 330.156 |
| A-n69-k9 | 69 | 9 | 1159.0 | 1518,94 | 1382,27 | 857.772 |
| B-n78-k10 | 78 | 10 | 1221.0 | 1784,30 | 1592.45 | 1335.523 |
| E-n101-k8 | 101 | 8 | 815.0 | 1824,06 | 1579.14 | 549.532 |
| M-121-k7 | 121 | 7 | 1034.0 | 3043,60 | 2868.68 | 723.965 |

## Anexo III: Pruebas de caja negra realizadas.

### Caso de prueba para el Caso de Uso “Cargar Instancia” (TSP) en TSVis.

##### Tabla 6: Clasificación de las condiciones de entrada para el caso de uso Cargar Instancia

|  |  |  |
| --- | --- | --- |
| **Condición de Entrada** | **Clases Válidas** | **Clases Inválidas** |
| Nombre del archivo | 1. Campo alfanumérico. | 1. Nulo (en blanco) 2. Archivo de estructura incorrecta. |

### Resultado de la ejecución de las clases de entrada propuestos.

##### Tabla 7: Resultados de la ejecución de las clases de entrada para el caso de uso Cargar Instancia

|  |  |  |
| --- | --- | --- |
| **Casos de Prueba** | **Clases** | **Resultado** |
| Nombre del archivo: “eil51.tsp” | 1 |  |
| Nombre del archivo: “ “ | 2 |  |
| Nombre del archivo: “eil51Incorrect.tsp” | 3 |  |

### Caso de prueba para el Caso de Uso “Ejecutar Búsqueda Tabú” (TSP) en TSVis.

##### Tabla 8: Clasificación de las condiciones de entrada para el caso de uso Ejecutar Búsqueda Tabú

|  |  |  |
| --- | --- | --- |
| **Condición de Entrada** | **Clases Válidas** | **Clases Inválidas** |
| *Diversification Criteria* | 1. Campo Numérico. |  |
| *Size TabuList* | 1. Campo Numérico. |  |
| *Neighborhood* | 1. *SwapMove* 2. *InsertMove* 3. *InvertMove* | 1. Nulo (en blanco) |
| *Graphics by Iterations* | 1. *On* 2. *Off* |  |

### Resultado de la ejecución de las clases de entrada propuestos.

##### Tabla 9: Resultados de la ejecución de las clases de entrada para el caso de uso Ejecutar Búsqueda Tabú

|  |  |  |
| --- | --- | --- |
| **Casos de Prueba** | **Clases** | **Resultado** |
| *Diversification Criteria*: “10”.  *Size TabuList*: “3”.  *Neighborhood*: “ “  *Graphics by Iterations: On* | 1, 2, 6, 7 |  |
| *Diversification Criteria*: “10”.  *Size TabuList*: “3”.  *Neighborhood: “SwapMove“*  *Graphics by Iterations: On* | 1, 2, 3, 7 | *Success* |

### Caso de prueba para el Caso de Uso Cargar Fichero en GAVis.

##### Tabla 10: Clasificación de las condiciones de entrada para el caso de uso Cargar Fichero.

|  |  |  |
| --- | --- | --- |
| Condición de Entrada | Clases Válidas | Clases No Válidas |
| Nombre de Archivo | 1. Campo Alfanumérico. | 1. Nulo (no selección) 2. Archivo de estructura incorrecta. |

### Resultado de la ejecución de las clases de entrada propuestos.

##### Tabla 11: Resultados de la ejecución de juegos de datos en el caso de uso Cargar Fichero.

|  |  |  |
| --- | --- | --- |
| Casos de Pruebas | Clases | Resultados |
| " " (no selección) | 2 |  |
| "TSPIncorrecto.tsp" | 3 |  |
| "att48.tsp" | 1 |  |

### Caso de prueba para el Caso de Uso Ejecutar Algoritmo Genético.

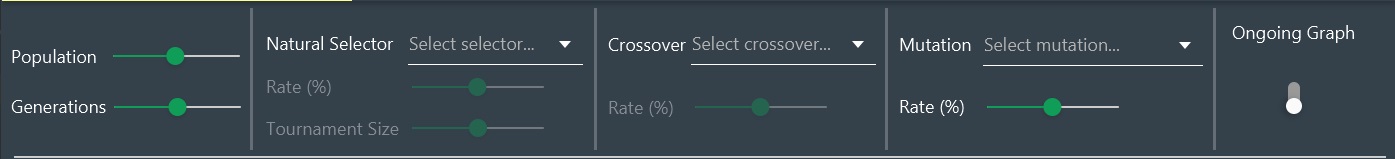


Ilustración 3: Selección de parámetros para la ejecución del Algoritmo Genético.

La siguiente tabla muestra un análisis de las condiciones de entrada, clases válidas y clases no válidas para el caso de uso **Ejecutar Algoritmo Genético**, específicamente en un problema TSP.

##### Tabla 12: Clasificación de las condiciones de entrada para el caso de uso Ejecutar Algoritmo Genético.

|  |  |  |
| --- | --- | --- |
| Condición de Entrada | Clases Válidas | Clases No Válidas |
| Population | 1. Campo Numérico. |  |
| Generations | 1. Campo Numérico. | 1. Cero (0) |
| Natural Selector | 1. Ranking. 2. Roulette. 3. Tournament. | 1. Nulo (no selección). |
| Rate (Natural Selector Ranking/Tournament) | 1. Campo Numérico. |  |
| Tournament Size (Natural Selector Tournament) | 1. Campo Numérico. |  |
| Crossover | 1. Cycle. 2. Ordered. 3. Grafenstettes. | 1. Nulo (no selección). |
| Rate (Crossover Cycle/Ordered) | 1. Campo Numérico. |  |
| Mutation | 1. Segment Swapping 2. Swapping | 1. Nulo (no selección). |
| Rate (Mutation Segment Swapping/Swapping) | 1. Campo Numérico. |  |
| Ongoing Graph | 1. Activado 2. Desactivado |  |

### Resultado de la ejecución de las clases de entrada propuestos.

##### Tabla 13: Resultados de la ejecución de juegos de datos en el caso de uso Ejecutar Algoritmo Genético.

|  |  |  |
| --- | --- | --- |
| Casos de Pruebas | Clases | Resultados |
| "250"  "0"  "Ranking"  "50"  "Cycle"  "50"  "Swapping"  "50"  Desactivado | 1, 3, 4, 8, 9, 10, 14, 16, 18, 20. |  |
| "250"  "5000"  "Tournament"  "75"  "15"  " "  "Swapping"  "50"  Activado | 1, 2, 6, 8, 9, 13, 16, 18, 19 |  |
| "250"  "4000"  "Ranking"  "50"  "Grafenstettes"  "Swapping"  "50"  Activado | 1, 2, 4, 8, 12, 16, 18, 20 | OK |

## Anexo IV: Representación de ficheros TSP y VRP.

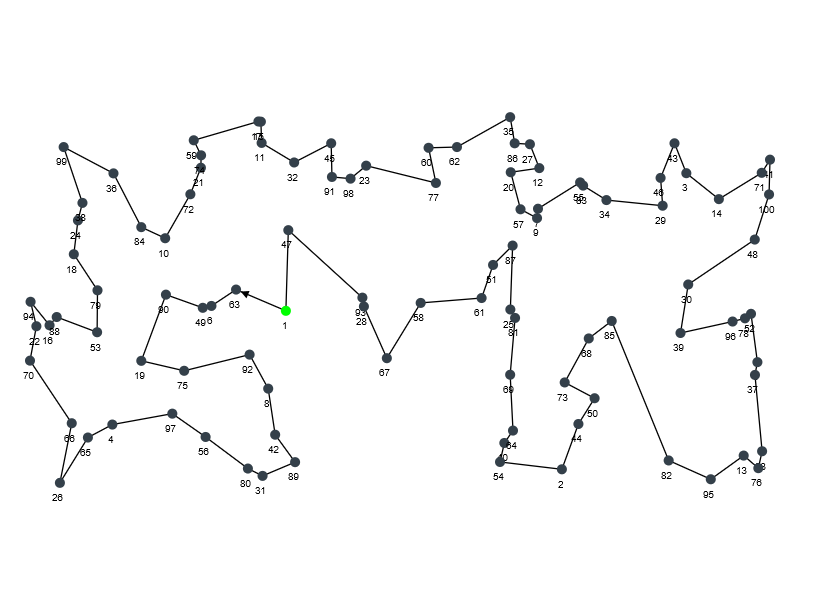


Ilustración 4: Solución con asistencia del usuario para el fichero kroA100.tsp usando TSVis (Resultado: 21282.0)

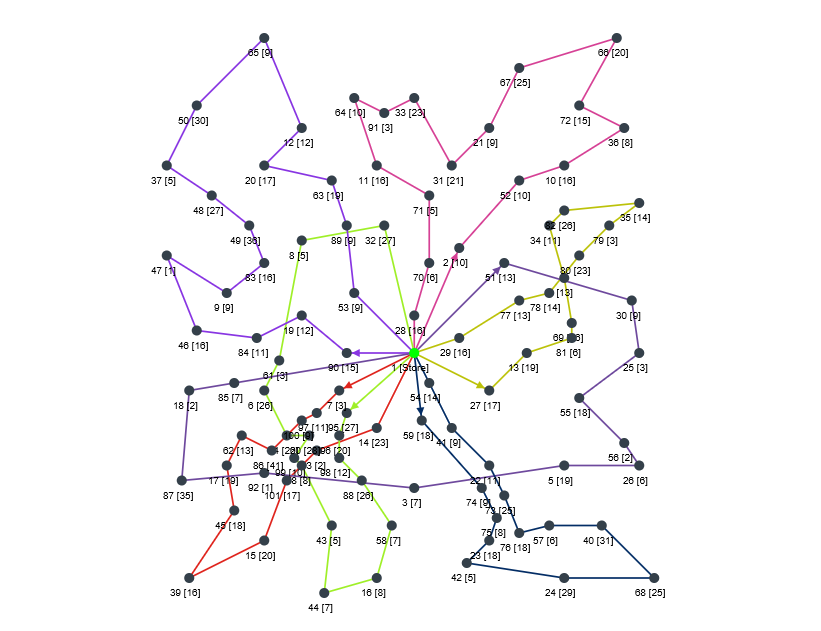


Ilustración 5: Solución con asistencia del usuario para el fichero E-n101-k8.vrp usando TSVis. (Resultado: 887.86)

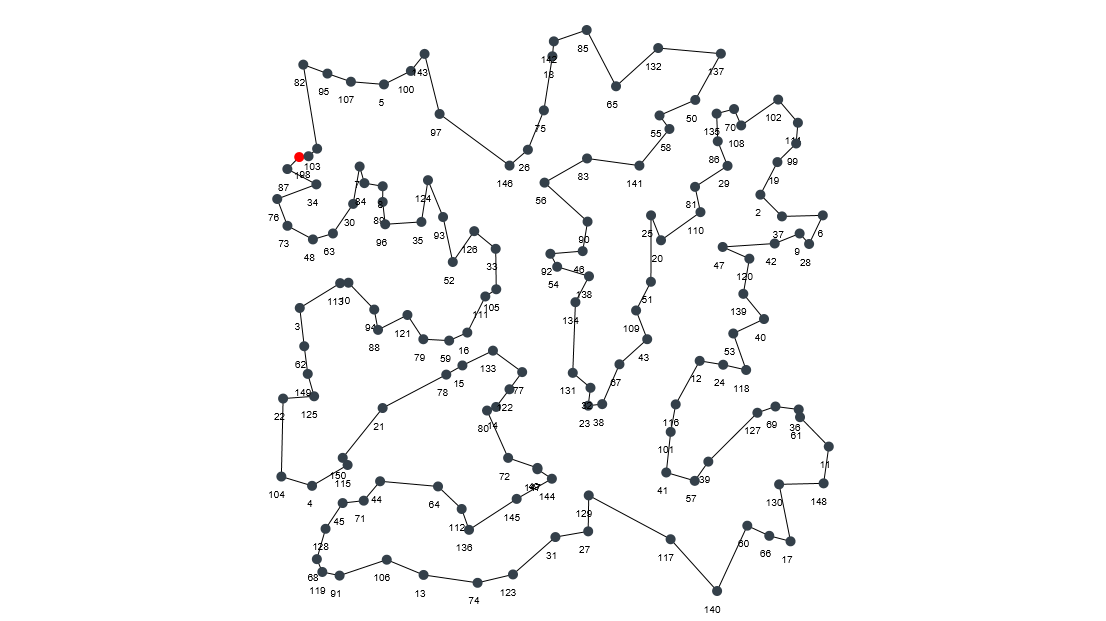


Ilustración 6: Solución con asistencia del usuario obtenida para el fichero ch150.tsp usando GAVis. (Resultado: 6567).

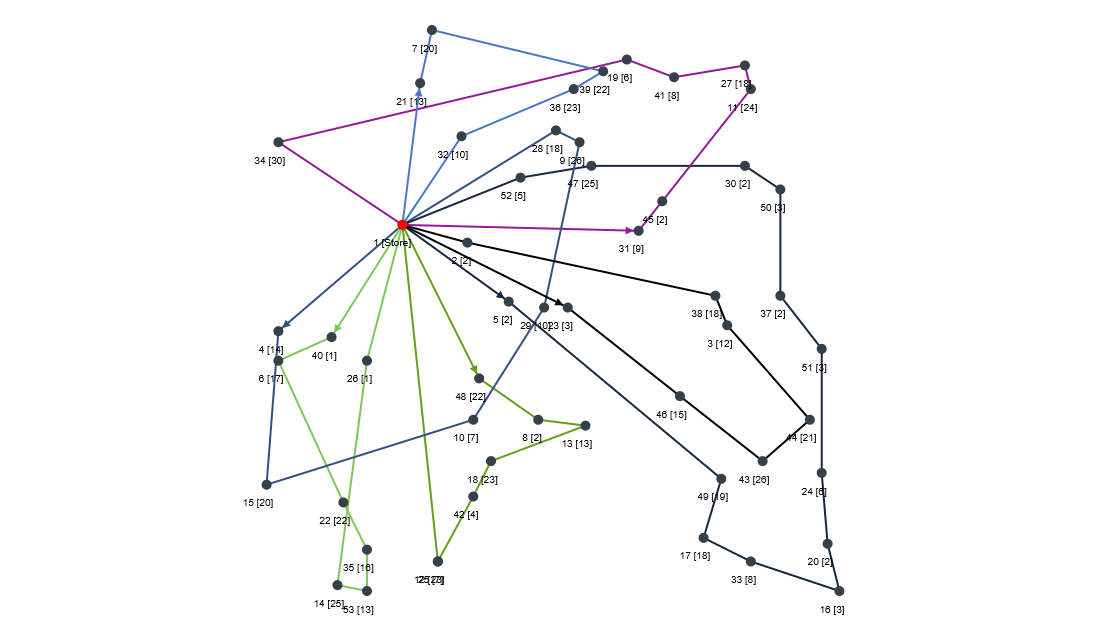


Ilustración 7: Solución con asistencia del usuario obtenida para el fichero A-n53-k7.vrp usando GAVis (Resultado: 1157,8).

1. Funciones que incorporan información adicional del problema utilizadas como estrategias para generar puntos de mejor calidad [↑](#footnote-ref-1)