Biased Random Key Genetic Algorithm con Búsqueda Local para el Team Orienteering Problem

Alejandro Lix Klett

Directora: Prof. Dra. Irene Loiseau

Departamento de Computación

June 6, 2018

Contenido I

Contenido II

Orienteering Problem

- Orientación es un deporte originario de Escandinavia

Orienteering Problem

- Orientación es un deporte originario de Escandinavia
- Cada jugador comienza en un punto de control y debe visitar tantos otros puntos de control como le sea posible dentro de un tiempo límite preespecificado.

Orienteering Problem

- Orientación es un deporte originario de Escandinavia
- Cada jugador comienza en un punto de control y debe visitar tantos otros puntos de control como le sea posible dentro de un tiempo límite preespecificado.
- Cada punto de control tiene un puntaje.

Orienteering Problem

- Orientación es un deporte originario de Escandinavia
- Cada jugador comienza en un punto de control y debe visitar tantos otros puntos de control como le sea posible dentro de un tiempo límite preespecificado.
- Cada punto de control tiene un puntaje.
- Cada punto de control puede ser visitado una sola vez a lo sumo.

Orienteering Problem

- Orientación es un deporte originario de Escandinavia
- Cada jugador comienza en un punto de control y debe visitar tantos otros puntos de control como le sea posible dentro de un tiempo límite preespecificado.
- Cada punto de control tiene un puntaje.
- Cada punto de control puede ser visitado una sola vez a lo sumo.
- El objetivo es maximizar el puntaje total.

Orienteering Problem

- Orientación es un deporte originario de Escandinavia
- Cada jugador comienza en un punto de control y debe visitar tantos otros puntos de control como le sea posible dentro de un tiempo límite preespecificado.
- Cada punto de control tiene un puntaje.
- Cada punto de control puede ser visitado una sola vez a lo sumo.
- El objetivo es maximizar el puntaje total.
- Este problema se conoce como Orienteering Problem (OP). El OP es NP-Hard como demostraron Golden, Levy y Vohra.

Team Orienteering Problem

- Hay M clientes, cada uno tiene un beneficio b_i y una coordenada en el plano.

Team Orienteering Problem

- Hay M clientes, cada uno tiene un beneficio b_i y una coordenada en el plano.
- Los puntos de salida y llegada tienen beneficio cero

Team Orienteering Problem

- Hay M clientes, cada uno tiene un beneficio b_i y una coordenada en el plano.
- Los puntos de salida y llegada tienen beneficio cero
- Hay N vehículos

Team Orienteering Problem

- Hay M clientes, cada uno tiene un beneficio b_i y una coordenada en el plano.
- Los puntos de salida y llegada tienen beneficio cero
- Hay N vehículos
- El beneficio de los clientes solo puede ser recolectado una vez.

Team Orienteering Problem

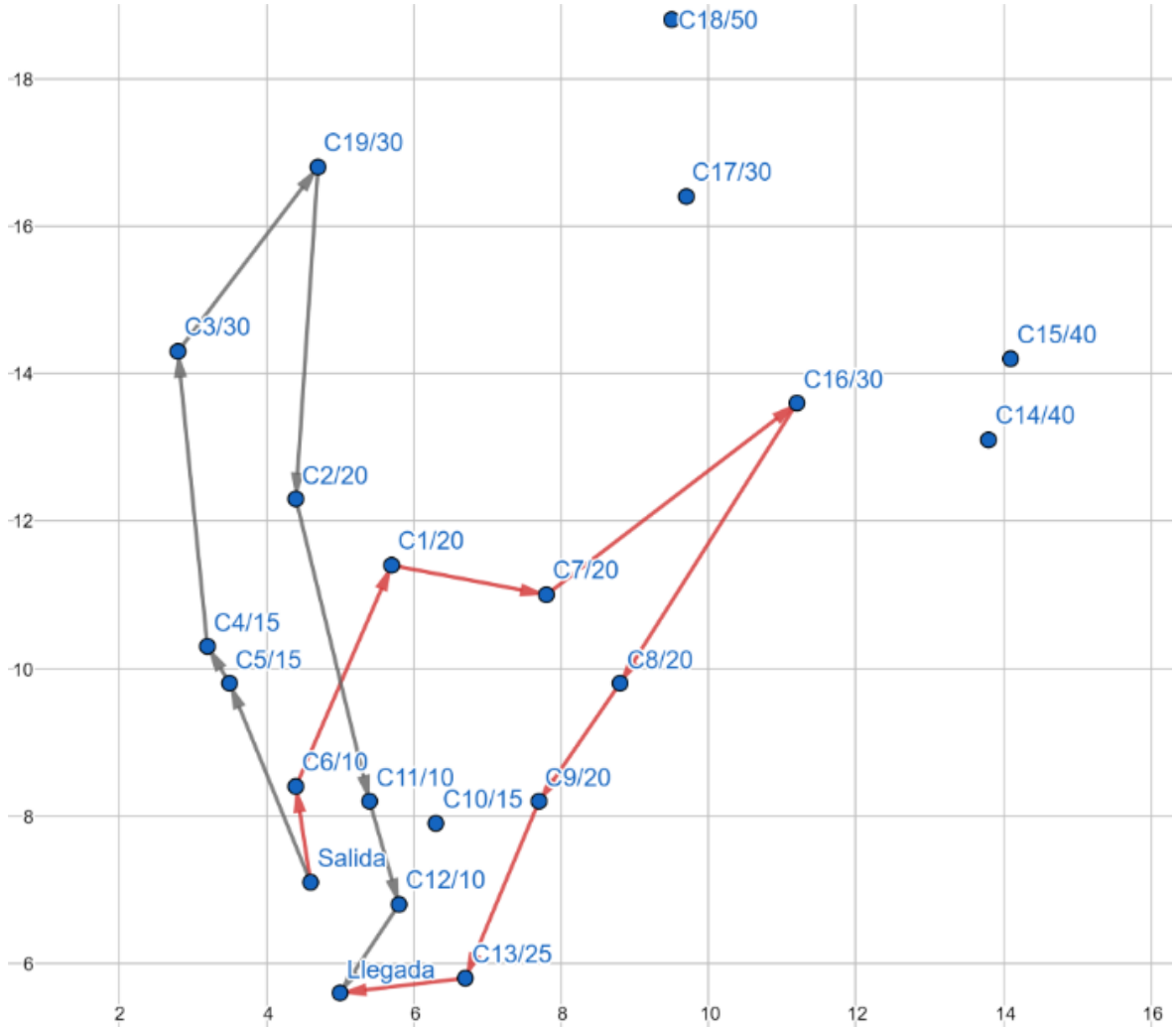
- Hay M clientes, cada uno tiene un beneficio b_i y una coordenada en el plano.
- Los puntos de salida y llegada tienen beneficio cero
- Hay N vehículos
- El beneficio de los clientes solo puede ser recolectado una vez.
- El objetivo es maximizar la sumatoria de los beneficios recolectados de todos los vehículos.

Team Orienteering Problem

- Hay M clientes, cada uno tiene un beneficio b_i y una coordenada en el plano.
- Los puntos de salida y llegada tienen beneficio cero
- Hay N vehículos
- El beneficio de los clientes solo puede ser recolectado una vez.
- El objetivo es maximizar la sumatoria de los beneficios recolectados de todos los vehículos.
- Como TOP contiene a OP, es al menos tan difícil.

Instancia p2.2.k del benchmark de Tsiligirides

La instancia tiene dos vehículos con un $d_{max} = 22{,}50$. Hay 19 clientes además de los puntos de salida y llegada.



Aplicaciones

- El deporte de orientación de equipo explicado anteriormente (que también se puede jugar en equipo).

Aplicaciones

- El deporte de orientación de equipo explicado anteriormente (que también se puede jugar en equipo).
- Tsiligirides hace mención del caso del Travelling Sales Person (TSP) sin tiempo suficiente para visitar a todos los clientes y que conoce de antemano el valor aproximado de las ventas que realizará en cada ciudad.

Aplicaciones

- El deporte de orientación de equipo explicado anteriormente (que también se puede jugar en equipo).
- Tsiligirides hace mención del caso del Travelling Sales Person (TSP) sin tiempo suficiente para visitar a todos los clientes y que conoce de antemano el valor aproximado de las ventas que realizará en cada ciudad.
- El reclutamiento de jugadores de fútbol americano universitario descrito en el trabajo de Butt y Cavalier. Los representante de las pequeñas ligas deben visitar la máxima cantidad de escuelas en un radio de 100 km, de antemano saben que no llegan a visitar a todas las escuelas.

Aplicaciones

- El problema de entrega de combustible con múltiples vehículos descrito en el trabajo Golden, Levy y Vohra. Una flota de camiones debe entregar combustible a una gran cantidad de clientes diariamente. El suministro de combustible del cliente debe mantenerse en un nivel adecuado en todo momento. Los desabastecimientos son costosos y deben evitarse en tanto sea posible.

Metaheurísticas

- Son métodos diseñados para encontrar buenas soluciones, en un tiempo razonable, a problemas de optimización combinatoria en general.

Metaheurísticas

- Son métodos diseñados para encontrar buenas soluciones, en un tiempo razonable, a problemas de optimización combinatoria en general.
- Las metaheurísticas son estrategias de alto nivel que guían una heurística específica del problema a resolver para mejorar su perfomance.

Metaheurísticas

- Son métodos diseñados para encontrar buenas soluciones, en un tiempo razonable, a problemas de optimización combinatoria en general.
- Las metaheurísticas son estrategias de alto nivel que guían una heurística específica del problema a resolver para mejorar su perfomance.

Características:

- Son estrategias que guían procesos de búsqueda.

Metaheurísticas

- Son métodos diseñados para encontrar buenas soluciones, en un tiempo razonable, a problemas de optimización combinatoria en general.
- Las metaheurísticas son estrategias de alto nivel que guían una heurística específica del problema a resolver para mejorar su perfomance.

Características:

- Son estrategias que guían procesos de búsqueda.
- Sus conceptos se pueden describir con un gran nivel de abstracción. No son para un problema específico.

Metaheurísticas

- Son métodos diseñados para encontrar buenas soluciones, en un tiempo razonable, a problemas de optimización combinatoria en general.
- Las metaheurísticas son estrategias de alto nivel que guían una heurística específica del problema a resolver para mejorar su perfomance.

Características:

- Son estrategias que guían procesos de búsqueda.
- Sus conceptos se pueden describir con un gran nivel de abstracción. No son para un problema específico.
- En muchos casos son algoritmos no-determinísticos.

Metaheurísticas

- Son métodos diseñados para encontrar buenas soluciones, en un tiempo razonable, a problemas de optimización combinatoria en general.
- Las metaheurísticas son estrategias de alto nivel que guían una heurística específica del problema a resolver para mejorar su perfomance.

Características:

- Son estrategias que guían procesos de búsqueda.
- Sus conceptos se pueden describir con un gran nivel de abstracción. No son para un problema específico.
- En muchos casos son algoritmos no-determinísticos.
- Sus desarrollos y diseños suelen estar motivados por comportamientos naturales.

Metaheurísticas

- Son métodos diseñados para encontrar buenas soluciones, en un tiempo razonable, a problemas de optimización combinatoria en general.
- Las metaheurísticas son estrategias de alto nivel que guían una heurística específica del problema a resolver para mejorar su perfomance.

Características:

- Son estrategias que guían procesos de búsqueda.
- Sus conceptos se pueden describir con un gran nivel de abstracción. No son para un problema específico.
- En muchos casos son algoritmos no-determinísticos.
- Sus desarrollos y diseños suelen estar motivados por comportamientos naturales.
- No garantizan que una solución óptima sea encontrada.

Metaheurísticas

- Son métodos diseñados para encontrar buenas soluciones, en un tiempo razonable, a problemas de optimización combinatoria en general.
- Las metaheurísticas son estrategias de alto nivel que guían una heurística específica del problema a resolver para mejorar su perfomance.

Características:

- Son estrategias que guían procesos de búsqueda.
- Sus conceptos se pueden describir con un gran nivel de abstracción. No son para un problema específico.
- En muchos casos son algoritmos no-determinísticos.
- Sus desarrollos y diseños suelen estar motivados por comportamientos naturales.
- No garantizan que una solución óptima sea encontrada.
- Las técnicas metaheurísticas van desde algoritmos simples de búsqueda local a complejos procesos de aprendizaje.

Metaheurísticas

Algunas técnicas:

- Simulated Annealing

Metaheurísticas

Algunas técnicas:

- Simulated Annealing
- Búsqueda Tabú

Metaheurísticas

Algunas técnicas:

- Simulated Annealing
- Búsqueda Tabú
- Algoritmos evolutivos

Metaheurísticas

Algunas técnicas:

- Simulated Annealing
- Búsqueda Tabú
- Algoritmos evolutivos
- Colonia de hormigas

Metaheurísticas

Algunas técnicas:

- Simulated Annealing
- Búsqueda Tabú
- Algoritmos evolutivos
- Colonia de hormigas
- Variable Neighborhood Search

Metaheurísticas

Algunas técnicas:

- Simulated Annealing
- Búsqueda Tabú
- Algoritmos evolutivos
- Colonia de hormigas
- Variable Neighborhood Search
- Iterated Local Search

Metaheurísticas

Algunas técnicas:

- Simulated Annealing
- Búsqueda Tabú
- Algoritmos evolutivos
- Colonia de hormigas
- Variable Neighborhood Search
- Iterated Local Search
- Etc

Algoritmos genéticos (GA)

- Motivados en el concepto de supervivencia del más apto.

Algoritmos genéticos (GA)

- Motivados en el concepto de supervivencia del más apto.
- Los algoritmos genéticos manejan un conjunto de individuos.

Algoritmos genéticos (GA)

- Motivados en el concepto de supervivencia del más apto.
- Los algoritmos genéticos manejan un conjunto de individuos.
- Cada individuo es un cromosoma que codifica una solución.

Algoritmos genéticos (GA)

- Motivados en el concepto de supervivencia del más apto.
- Los algoritmos genéticos manejan un conjunto de individuos.
- Cada individuo es un cromosoma que codifica una solución.
- Cada cromosoma tiene asociado un nivel de condición física que está correlacionado con el correspondiente valor de la función objetivo de la solución que codifica.

Algoritmos genéticos (GA)

- Motivados en el concepto de supervivencia del más apto.
- Los algoritmos genéticos manejan un conjunto de individuos.
- Cada individuo es un cromosoma que codifica una solución.
- Cada cromosoma tiene asociado un nivel de condición física que está correlacionado con el correspondiente valor de la función objetivo de la solución que codifica.
- En cada generación se crea una nueva población con individuos provenientes de tres fuentes distintas: crossover, elites y mutantes.

Random Key Genetic Algorithm (RKGA)

- Los individuos son representados por un vector de números reales en el intervalo [0, 1].

Random Key Genetic Algorithm (RKGA)

- Los individuos son representados por un vector de números reales en el intervalo [0, 1].
- La población inicial es generada al azar.

Random Key Genetic Algorithm (RKGA)

- Los individuos son representados por un vector de números reales en el intervalo [0, 1].
- La población inicial es generada al azar.
- El decodificador es el responsable de convertir un cromosoma en una solución válida del problema.

Random Key Genetic Algorithm (RKGA)

- Los individuos son representados por un vector de números reales en el intervalo [0, 1].
- La población inicial es generada al azar.
- El decodificador es el responsable de convertir un cromosoma en una solución válida del problema.
- En cada iteración se toman los mejores individuos y pasan directamente a la siguiente generación (elites).

Random Key Genetic Algorithm (RKGA)

- Los individuos son representados por un vector de números reales en el intervalo [0, 1].
- La población inicial es generada al azar.
- El decodificador es el responsable de convertir un cromosoma en una solución válida del problema.
- En cada iteración se toman los mejores individuos y pasan directamente a la siguiente generación (elites).
- La mayoría de los individuos de la nueva generación se genera cruzando dos individuos de la generación actual (crossover).

Random Key Genetic Algorithm (RKGA)

- Los individuos son representados por un vector de números reales en el intervalo [0, 1].
- La población inicial es generada al azar.
- El decodificador es el responsable de convertir un cromosoma en una solución válida del problema.
- En cada iteración se toman los mejores individuos y pasan directamente a la siguiente generación (elites).
- La mayoría de los individuos de la nueva generación se genera cruzando dos individuos de la generación actual (crossover).
- Un porcentaje muy bajo de los nuevos individuos es generado al azar, para escapar de mínimos locales (mutantes).

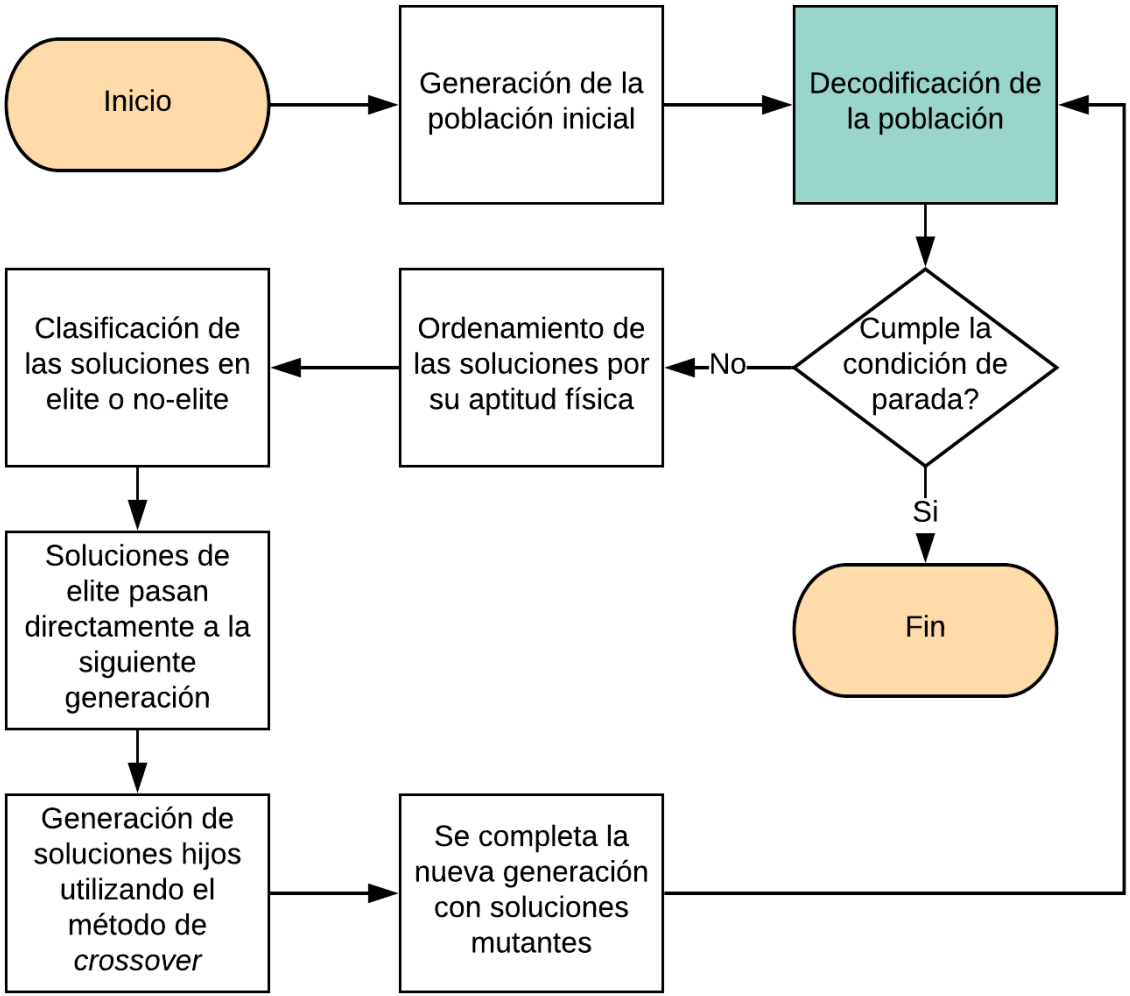
Biased Random Key Genetic Algorithm (BRKGA)

- Cada individuo se genera combinando un elemento seleccionado al azar del conjunto de elite y el otro de la conjunto no elite.

Biased Random Key Genetic Algorithm (BRKGA)

- Cada individuo se genera combinando un elemento seleccionado al azar del conjunto de elite y el otro de la conjunto no elite.
- Parameterized Uniform Crossover. La probabilidad de que se trasmita el alelo del padre de elite es mayor que la del padre de no elite.

Diagrama de flujo del BRKGA



Generación de la población inicial

- Se crea una cantidad de vectores de enteros aleatorios igual a la cantidad de soluciones por generación que se desea.

Generación de la población inicial

- Se crea una cantidad de vectores de enteros aleatorios igual a la cantidad de soluciones por generación que se desea.
- Los vectores tienen un tamaño igual a la cantidad de clientes de la instancia.

Generación de la población inicial

- Se crea una cantidad de vectores de enteros aleatorios igual a la cantidad de soluciones por generación que se desea.
- Los vectores tienen un tamaño igual a la cantidad de clientes de la instancia.
- Cada entero aleatorio del vector esta asociado a un identificador de cliente.

Generación de la población inicial

- Se crea una cantidad de vectores de enteros aleatorios igual a la cantidad de soluciones por generación que se desea.
- Los vectores tienen un tamaño igual a la cantidad de clientes de la instancia.
- Cada entero aleatorio del vector esta asociado a un identificador de cliente.

Ejemplo de un nuevo vector de enteros aleatorios.

Key	27	13	79	45	21	7	98	54
ClientId	1	2	3	4	5	6	7	8

Decodificación de los vectores en soluciones válidas del problema

- Se ordena el vector de enteros aleatorios por el valor de la clave aleatoria de forma ascendente.

Decodificación de los vectores en soluciones válidas del problema

- Se ordena el vector de enteros aleatorios por el valor de la clave aleatoria de forma ascendente.

Ejemplo del vector de enteros aleatorios ordenado.

Key	7	13	21	27	45	54	79	98
ClientId	6	2	5	1	4	8	3	7

Decodificación de los vectores en soluciones válidas del problema

- Se ordena el vector de enteros aleatorios por el valor de la clave aleatoria de forma ascendente.

Ejemplo del vector de enteros aleatorios ordenado.

Key	7	13	21	27	45	54	79	98
ClientId	6	2	5	1	4	8	3	7

- Implementé dos decodificadores cada uno con su estrategia para generar soluciones.

Decodificación de los vectores en soluciones válidas del problema

- Se ordena el vector de enteros aleatorios por el valor de la clave aleatoria de forma ascendente.

Ejemplo del vector de enteros aleatorios ordenado.

Key	7	13	21	27	45	54	79	98
ClientId	6	2	5	1	4	8	3	7

- Implementé dos decodificadores cada uno con su estrategia para generar soluciones.
- Ambos decodificadores generan una solución válida del problema a partir de un vector de enteros aleatorios ordenado.

Decodificador simple

- Los vehículos están ordenados de forma ascendente según su identificador.

Decodificador simple

- Los vehículos están ordenados de forma ascendente según su identificador.
- Toma el primer cliente e intenta agregarlo en la ruta del primer vehículo disponible.

Decodificador simple

- Los vehículos están ordenados de forma ascendente según su identificador.
- Toma el primer cliente e intenta agregarlo en la ruta del primer vehículo disponible.
- Si logra insertarlo repite el proceso con el siguiente cliente para el mismo vehículo.

Decodificador simple

- Los vehículos están ordenados de forma ascendente según su identificador.
- Toma el primer cliente e intenta agregarlo en la ruta del primer vehículo disponible.
- Si logra insertarlo repite el proceso con el siguiente cliente para el mismo vehículo.
- Si no lo logra, considera que la ruta del vehículo actual esta completa e intenta agregar el mismo cliente en el siguiente vehículo disponible.

Decodificador simple

- Los vehículos están ordenados de forma ascendente según su identificador.
- Toma el primer cliente e intenta agregarlo en la ruta del primer vehículo disponible.
- Si logra insertarlo repite el proceso con el siguiente cliente para el mismo vehículo.
- Si no lo logra, considera que la ruta del vehículo actual esta completa e intenta agregar el mismo cliente en el siguiente vehículo disponible.
- Repite hasta completar la ruta de todos los vehículos disponibles.

Decodificador simple

- Los vehículos están ordenados de forma ascendente según su identificador.
- Toma el primer cliente e intenta agregarlo en la ruta del primer vehículo disponible.
- Si logra insertarlo repite el proceso con el siguiente cliente para el mismo vehículo.
- Si no lo logra, considera que la ruta del vehículo actual esta completa e intenta agregar el mismo cliente en el siguiente vehículo disponible.
- Repite hasta completar la ruta de todos los vehículos disponibles.

Ejemplo de la solución generada por el decodificador simple

Key	7	13	21	27	45	54	79	89
ClientId	6	2	5	1	4	8	3	7

Vehículo 1: 6 -> 2
Vehículo 2: 5 -> 1

Decodificador goloso

- Se diferencia del decodificador simple en el momento en que encuentra un cliente que no entra en la ruta del vehículo actual.

Decodificador goloso

- Se diferencia del decodificador simple en el momento en que encuentra un cliente que no entra en la ruta del vehículo actual.
- En vez de pasar al siguiente vehículo, prueba con el siguiente cliente.

Decodificador goloso

- Se diferencia del decodificador simple en el momento en que encuentra un cliente que no entra en la ruta del vehículo actual.
- En vez de pasar al siguiente vehículo, prueba con el siguiente cliente.
- Por lo tanto, por cada vehículo prueba todos los clientes en el orden dado.

Decodificador goloso

- Se diferencia del decodificador simple en el momento en que encuentra un cliente que no entra en la ruta del vehículo actual.
- En vez de pasar al siguiente vehículo, prueba con el siguiente cliente.
- Por lo tanto, por cada vehículo prueba todos los clientes en el orden dado.
- No prueba con los clientes que ya fueron asignados a otro vehículo.

Decodificador goloso

- Se diferencia del decodificador simple en el momento en que encuentra un cliente que no entra en la ruta del vehículo actual.
- En vez de pasar al siguiente vehículo, prueba con el siguiente cliente.
- Por lo tanto, por cada vehículo prueba todos los clientes en el orden dado.
- No prueba con los clientes que ya fueron asignados a otro vehículo.

Ejemplo de la solución generada por el decodificador goloso

Key	7	13	21	27	45	54	79	89
ClientId	6	2	5	1	4	8	3	7

Vehículo 1: 6 -> 2 -> 8
Vehículo 2: 5 -> 1 -> 3

Descripción de las columnas de resultados

- *Instancia*: Nombre de la instancia utilizada.
- $N/V/D$: Cantidad de **N**odos / Cantidad de **V**ehículos / **D**istancia máxima de la ruta del vehículo
- T_{avg}: El **T**iempo promedio en milisegundos de la ejecución del algoritmo para la instancia mencionada sobre n ejecuciones.
- D: El **D**ecodificador utilizado.

Descripción de las columnas de resultados

- B_{max}: El **B**eneficio máximo que obtuve para la instancia mencionada sobre n ejecuciones.
- B_{min}: El **B**eneficio mínimo que obtuve para la instancia mencionada sobre n ejecuciones.
- B_{avg}: El **B**eneficio promedio que obtuve para la instancia mencionada sobre n ejecuciones.
- i_{eMax}: Indice de efectividad máximo. $i_{eMax} = B_{max}/Best$
- i_{eAvg}: Indice de efectividad promedio. $i_{eAvg} = B_{avg}/Best$
- $Best$: Máximo beneficio obtenido por algún trabajo previo sobre la misma instancia mencionada.

Comparación entre decodificadores

- Se realizó una prueba para comparar los tiempos de ejecución y aptitud de las soluciones generadas por ambos decodificadores.

Comparación entre decodificadores

- Se realizó una prueba para comparar los tiempos de ejecución y aptitud de las soluciones generadas por ambos decodificadores.
- Se crearon 200 vectores de enteros aleatorios y se decodificaron utilizando ambos decodificadores.

Comparación entre decodificadores

- Se realizó una prueba para comparar los tiempos de ejecución y aptitud de las soluciones generadas por ambos decodificadores.
- Se crearon 200 vectores de enteros aleatorios y se decodificaron utilizando ambos decodificadores.

Instancia	N/V/D	D	B_{min}	B_{avg}	B_{max}	i_{eAvg}	*Best*
p2.2.k	21/2/22.50	S	50	101	170	0.37	275
p2.2.k	21/2/22.50	G	105	166	230	0.60	275
p2.3.g	21/3/10.70	S	45	84	140	0.58	145
p2.3.g	21/3/10.70	G	90	122	145	0.84	145
p5.3.x	66/3/40.00	S	195	301	425	0.19	1555
p5.3.x	66/3/40.00	G	305	412	560	0.26	1555
p7.2.e	102/2/50.00	S	8	39	113	0.13	290
p7.2.e	102/2/50.00	G	37	95	171	0.33	290
p7.4.t	102/4/100.00	S	38	114	238	0.11	1077
p7.4.t	102/4/100.00	G	165	273	439	0.25	1077

Condición de parada

- Cantidad de iteraciones.

Condición de parada

- Cantidad de iteraciones.
- Últimas X generaciónes sin que haya mejorado el beneficio de la mejor solución.

Condición de parada

- Cantidad de iteraciones.
- Últimas X generaciónes sin que haya mejorado el beneficio de la mejor solución.
- Ambas condiciones deben ser válidas a la vez para que termine el algoritmo.

Evolución de la población

- Se ordenan las soluciones por su aptitud física. (vectores decodificados)

Evolución de la población

- Se ordenan las soluciones por su aptitud física. (vectores decodificados)
- Las mejores X soluciones se agregan al conjunto elite y pasan directamente a la siguiente generación.

Evolución de la población

- Se ordenan las soluciones por su aptitud física. (vectores decodificados)
- Las mejores X soluciones se agregan al conjunto elite y pasan directamente a la siguiente generación.
- El resto de las $\#Poblacion - X$ soluciones, se agregan al conjunto no elite.

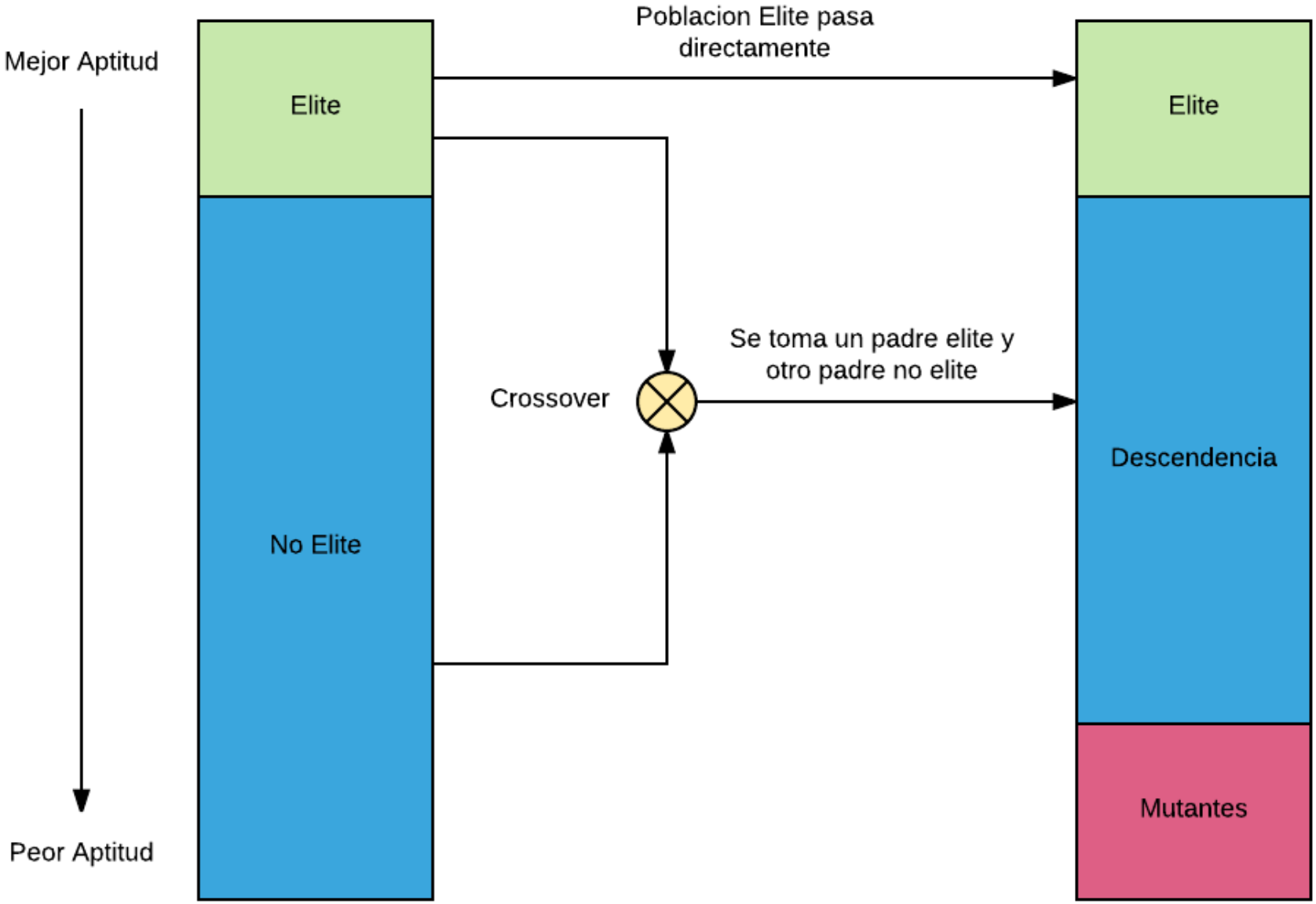
Evolución de la población

- Se ordenan las soluciones por su aptitud física. (vectores decodificados)
- Las mejores X soluciones se agregan al conjunto elite y pasan directamente a la siguiente generación.
- El resto de las $\#Poblacion - X$ soluciones, se agregan al conjunto no elite.
- Se generan Y soluciones mutantes del mismo modo que se generaron las soluciones iniciales que se agregan a la nueva generación.

Evolución de la población

- Se ordenan las soluciones por su aptitud física. (vectores decodificados)
- Las mejores X soluciones se agregan al conjunto elite y pasan directamente a la siguiente generación.
- El resto de las $\#Poblacion - X$ soluciones, se agregan al conjunto no elite.
- Se generan Y soluciones mutantes del mismo modo que se generaron las soluciones iniciales que se agregan a la nueva generación.
- Se completa la nueva generación mediante el proceso de *crossover*. Proceso por el cual a partir de dos individuos se genera un tercer individuo.

Evolución de la población



Crossover

- Se toma un cromosoma del conjunto de elite y otro del conjunto de no elite.

Crossover

- Se toma un cromosoma del conjunto de elite y otro del conjunto de no elite.
- Se crea un vector de números reales al azar en el intervalo [0, 1] del mismo tamaño que los cromosomas.

Crossover

- Se toma un cromosoma del conjunto de elite y otro del conjunto de no elite.
- Se crea un vector de números reales al azar en el intervalo [0, 1] del mismo tamaño que los cromosomas.
- Si el valor en la posición i del vector de números reales es menor a ρ_e, el cromosoma hijo hereda el alelo de la posición i del padre elite. En caso contrario el cromosoma hijo hereda el alelo de la posición i del padre no elite.

Crossover

- Se toma un cromosoma del conjunto de elite y otro del conjunto de no elite.
- Se crea un vector de números reales al azar en el intervalo [0, 1] del mismo tamaño que los cromosomas.
- Si el valor en la posición i del vector de números reales es menor a ρ_e, el cromosoma hijo hereda el alelo de la posición i del padre elite. En caso contrario el cromosoma hijo hereda el alelo de la posición i del padre no elite.
- El valor de ρ_e suele ser 0.70, favoreciendo los alelos del cromosoma elite.

Crossover


Cromosoma Elite
0.17
0.48
0.25
0.89
0.68
Cromosoma No Elite
0.28
0.72
0.53
0.61
0.11
Números al Azar
0.73
0.41
0.57
0.11
0.81
Relación (0.70)
>
<
<
<
>
Cromosoma Resultante
0.28
0.48
0.25
0.89
0.11

Hash de un individuo

- Dos individuos son iguales si representan la misma solución.

Hash de un individuo

- Dos individuos son iguales si representan la misma solución.
- Tener individuos repetidos en la población reduce el dominio de soluciones exploradas.

Hash de un individuo

- Dos individuos son iguales si representan la misma solución.
- Tener individuos repetidos en la población reduce el dominio de soluciones exploradas.
- Tener individuos repetidos en la población genera cálculos repetidos.

Hash de un individuo

- Dos individuos son iguales si representan la misma solución.
- Tener individuos repetidos en la población reduce el dominio de soluciones exploradas.
- Tener individuos repetidos en la población genera cálculos repetidos.
- Un individuo se inserta en la población si no existe algún individuo en la población con el mismo valor de hash.

Hash de un individuo

- Dos individuos son iguales si representan la misma solución.
- Tener individuos repetidos en la población reduce el dominio de soluciones exploradas.
- Tener individuos repetidos en la población genera cálculos repetidos.
- Un individuo se inserta en la población si no existe algún individuo en la población con el mismo valor de hash.
- Se implementaron dos formas de calcular el hash de un individuo.

Hash de un individuo

- Dos individuos son iguales si representan la misma solución.
- Tener individuos repetidos en la población reduce el dominio de soluciones exploradas.
- Tener individuos repetidos en la población genera cálculos repetidos.
- Un individuo se inserta en la población si no existe algún individuo en la población con el mismo valor de hash.
- Se implementaron dos formas de calcular el hash de un individuo.
- La primer forma que implemente calcula el hash sin conocer el problema que se esta resolviendo, luego mantiene la evolución de la población independiente del problema que se esta resolviendo.

Hash de un individuo

- Dos individuos son iguales si representan la misma solución.
- Tener individuos repetidos en la población reduce el dominio de soluciones exploradas.
- Tener individuos repetidos en la población genera cálculos repetidos.
- Un individuo se inserta en la población si no existe algún individuo en la población con el mismo valor de hash.
- Se implementaron dos formas de calcular el hash de un individuo.
- La primer forma que implemente calcula el hash sin conocer el problema que se esta resolviendo, luego mantiene la evolución de la población independiente del problema que se esta resolviendo.
- La segunda forma de calcular es óptima detectando repetidos y requiere conocer el problema que se esta resolviendo.

Método independiente del problema para calcular el hash

- Se toma el vector de claves aleatorias del individuo y se lo ordena por su clave aleatoria. El valor de su hash es la concatenación de los identificadores de clientes separados con un símbolo.

Método independiente del problema para calcular el hash

- Se toma el vector de claves aleatorias del individuo y se lo ordena por su clave aleatoria. El valor de su hash es la concatenación de los identificadores de clientes separados con un símbolo.

Key	27	13	79	45	21	7	98	54
ClientId	1	2	3	4	5	6	7	8

Key	7	13	21	27	45	54	79	98
ClientId	6	2	5	1	4	8	3	7

Método independiente del problema para calcular el hash

- Se toma el vector de claves aleatorias del individuo y se lo ordena por su clave aleatoria. El valor de su hash es la concatenación de los identificadores de clientes separados con un símbolo.

Key	27	13	79	45	21	7	98	54
ClientId	1	2	3	4	5	6	7	8

Key	7	13	21	27	45	54	79	98
ClientId	6	2	5	1	4	8	3	7

- Hash resultante del vector de enteros de la imagen: 6@2@5@1@4@8@3@7

Método óptimo para calcular el hash

- Se toman los recorridos de los vehículos y se los ordena por el identificador del primer cliente que visitan.

Método óptimo para calcular el hash

- Se toman los recorridos de los vehículos y se los ordena por el identificador del primer cliente que visitan.
- Se calcula el hash de cada vehículo y se concatenan utilizando otro símbolo separador.

Método óptimo para calcular el hash

- Se toman los recorridos de los vehículos y se los ordena por el identificador del primer cliente que visitan.
- Se calcula el hash de cada vehículo y se concatenan utilizando otro símbolo separador.

Vector de enteros aleatorios ordenado y la solución en la que decodifica.

Key	7	13	21	27	45	54	79	89
ClientId	6	2	5	1	4	8	3	7

Vehículo 1: 6 -> 2
Vehículo 2: 5 -> 1

Método óptimo para calcular el hash

- Se toman los recorridos de los vehículos y se los ordena por el identificador del primer cliente que visitan.
- Se calcula el hash de cada vehículo y se concatenan utilizando otro símbolo separador.

Vector de enteros aleatorios ordenado y la solución en la que decodifica.

Key	7	13	21	27	45	54	79	89
ClientId	6	2	5	1	4	8	3	7

Vehículo 1: 6 -> 2
Vehículo 2: 5 -> 1

- Hash resultante de la solución de la imagen: 5@1#6@2

Método óptimo para calcular el hash

- Se toman los recorridos de los vehículos y se los ordena por el identificador del primer cliente que visitan.
- Se calcula el hash de cada vehículo y se concatenan utilizando otro símbolo separador.

Vector de enteros aleatorios ordenado y la solución en la que decodifica.

Key	7	13	21	27	45	54	79	89
ClientId	6	2	5	1	4	8	3	7

Vehículo 1: 6 -> 2
Vehículo 2: 5 -> 1

- Hash resultante de la solución de la imagen: 5@1#6@2
- Si cambiamos la posición de los ClientId 8, 3 y 7 obtenemos la misma solución al decodificarla. El hash con este método no cambia y el hash con el método anterior cambia.

Pruebas sobre el BRKGA sin búsqueda local

- Se realizaron varias pruebas sobre el BRKGA modificando las variables de su configuración. Tales variables:

Pruebas sobre el BRKGA sin búsqueda local

- Se realizaron varias pruebas sobre el BRKGA modificando las variables de su configuración. Tales variables:
- Iteraciones. (Entre 200 y 300).

Pruebas sobre el BRKGA sin búsqueda local

- Se realizaron varias pruebas sobre el BRKGA modificando las variables de su configuración. Tales variables:
- Iteraciones. (Entre 200 y 300).
- Cantidad de iteraciones sin cambios en la mejor solución. (Entre 50 y 100).

Pruebas sobre el BRKGA sin búsqueda local

- Se realizaron varias pruebas sobre el BRKGA modificando las variables de su configuración. Tales variables:
- Iteraciones. (Entre 200 y 300).
- Cantidad de iteraciones sin cambios en la mejor solución. (Entre 50 y 100).
- Tamaño de la población. (Entre 100 y 200).

Pruebas sobre el BRKGA sin búsqueda local

- Se realizaron varias pruebas sobre el BRKGA modificando las variables de su configuración. Tales variables:
- Iteraciones. (Entre 200 y 300).
- Cantidad de iteraciones sin cambios en la mejor solución. (Entre 50 y 100).
- Tamaño de la población. (Entre 100 y 200).
- Porcentaje de la población elite. (Entre el 20% y el 30%)

Pruebas sobre el BRKGA sin búsqueda local

- Se realizaron varias pruebas sobre el BRKGA modificando las variables de su configuración. Tales variables:
- Iteraciones. (Entre 200 y 300).
- Cantidad de iteraciones sin cambios en la mejor solución. (Entre 50 y 100).
- Tamaño de la población. (Entre 100 y 200).
- Porcentaje de la población elite. (Entre el 20% y el 30%)
- Porcentaje de mutantes. (Entre el 5% y el 10%)

Pruebas sobre el BRKGA sin búsqueda local

- Se realizaron varias pruebas sobre el BRKGA modificando las variables de su configuración. Tales variables:
- Iteraciones. (Entre 200 y 300).
- Cantidad de iteraciones sin cambios en la mejor solución. (Entre 50 y 100).
- Tamaño de la población. (Entre 100 y 200).
- Porcentaje de la población elite. (Entre el 20% y el 30%)
- Porcentaje de mutantes. (Entre el 5% y el 10%)
- Probabilidad de heredar alelo del padre elite ($\rho_e \in [0.5, 0.7]$)

Pruebas sobre el BRKGA sin búsqueda local

- Se realizaron varias pruebas sobre el BRKGA modificando las variables de su configuración. Tales variables:
- Iteraciones. (Entre 200 y 300).
- Cantidad de iteraciones sin cambios en la mejor solución. (Entre 50 y 100).
- Tamaño de la población. (Entre 100 y 200).
- Porcentaje de la población elite. (Entre el 20% y el 30%)
- Porcentaje de mutantes. (Entre el 5% y el 10%)
- Probabilidad de heredar alelo del padre elite ($\rho_e \in [0.5, 0.7]$)
- Tipo de decodificador. (El simple o el goloso).

La configuración que mejor resulto para el BRKGA sin Búsqueda Local

- Luego de varias pruebas, evaluando los resultados obtenidos y el tiempo de ejecución, la configuración que obtuvo los mejores resultados:

La configuración que mejor resulto para el BRKGA sin Búsqueda Local

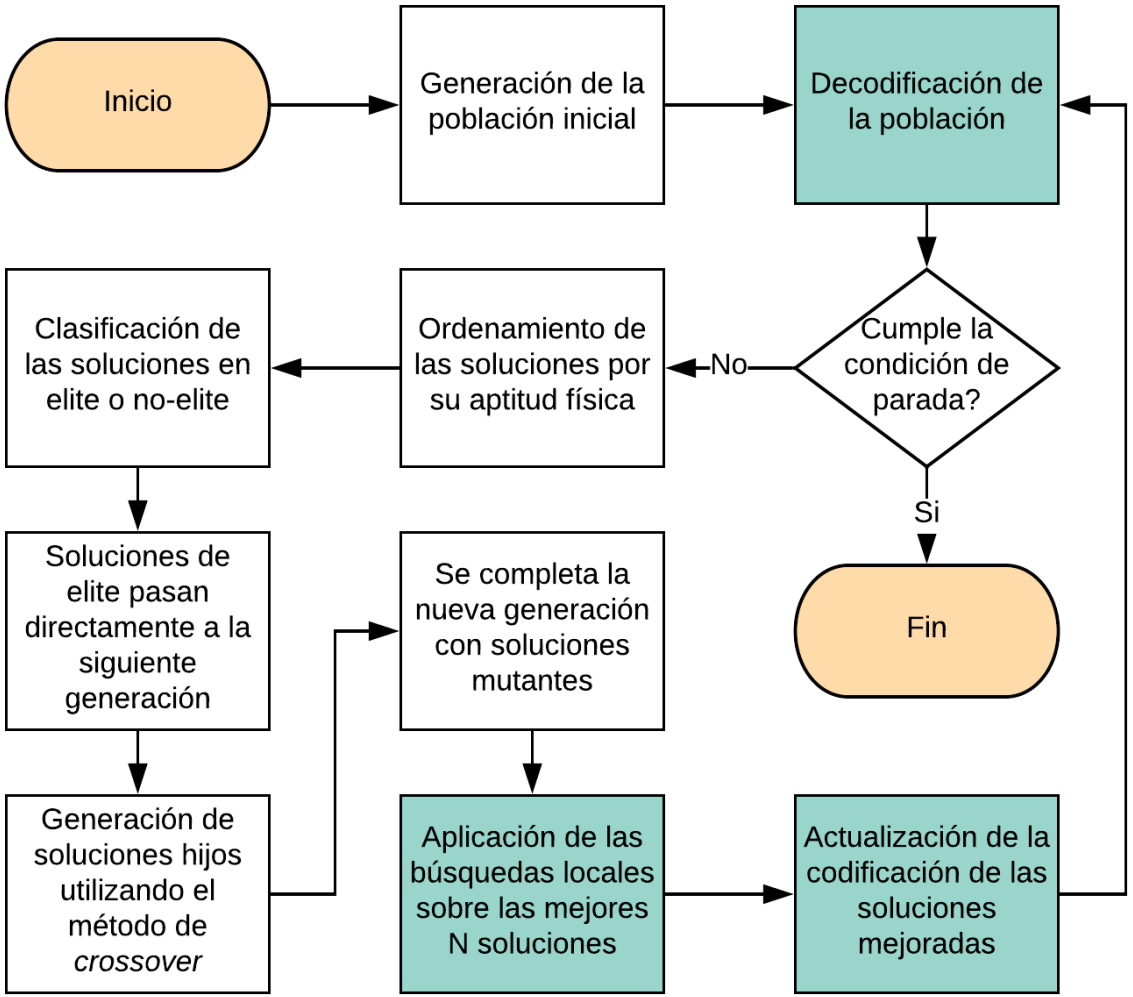
- Luego de varias pruebas, evaluando los resultados obtenidos y el tiempo de ejecución, la configuración que obtuvo los mejores resultados:
- Iteraciones: 250; Sin cambios: 50; Población.100; Porcentaje elite: 30%; Porcentaje mutante: 10%; ρ_e: 0.70; Deco: Goloso.

La configuración que mejor resulto para el BRKGA sin Búsqueda Local

- Luego de varias pruebas, evaluando los resultados obtenidos y el tiempo de ejecución, la configuración que obtuvo los mejores resultados:
- Iteraciones: 250; Sin cambios: 50; Población.100; Porcentaje elite: 30%; Porcentaje mutante: 10%; ρ_e: 0.70; Deco: Goloso.
- La tabla muestra los resultados de 10 ejecuciones por instancia.

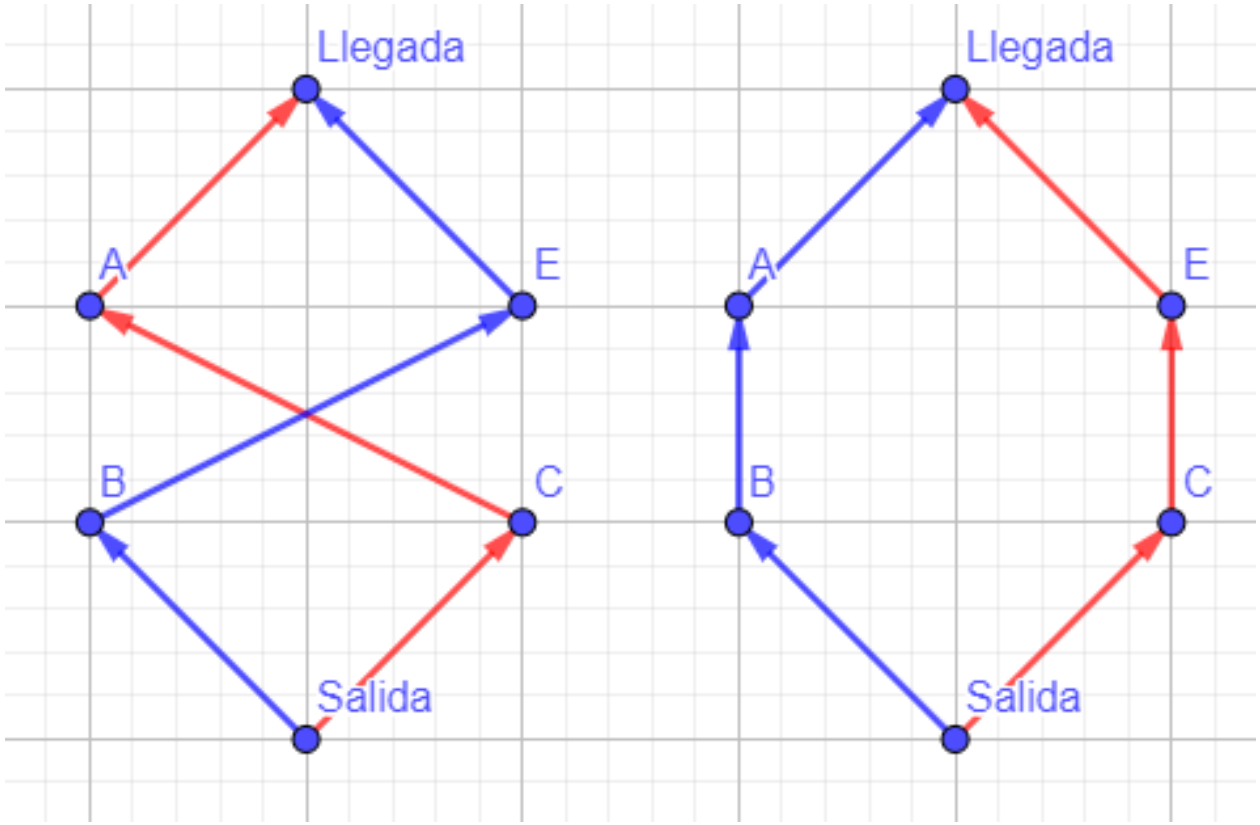
Instancia	N/V/D	B_{min}	B_{avg}	B_{max}	i_{eAvg}	i_{eMax}	*Best*
p2.2.k	21/2/22.50	240	253	260	0.92	0.95	275
p2.3.g	21/3/10.70	145	145	145	1.00	1.00	145
p3.4.p	33/4/22.50	420	435	450	0.78	0.80	560
p5.3.x	66/3/40.00	735	735	735	0.47	0.47	1555
p7.2.e	102/2/50.00	200	209	222	0.72	0.77	290
p7.4.t	102/4/100.00	461	478	505	0.44	0.47	1077

Flow chart del BRKGA con búsqueda local



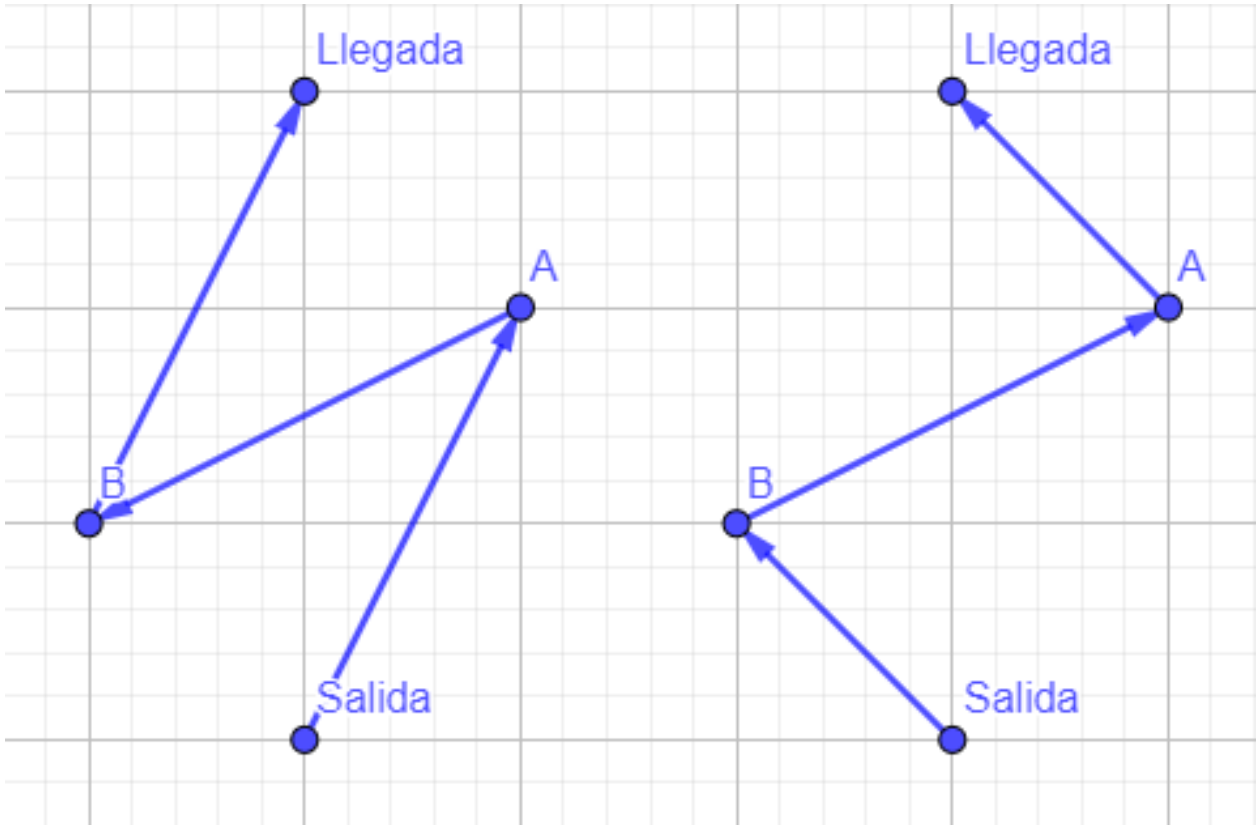
Búsqueda local: Swap

- Dados dos vehículos, busca intercambiar clientes entre sus rutas con el objetivo de reducir la suma de las distancias recorridas por ambos vehículos.



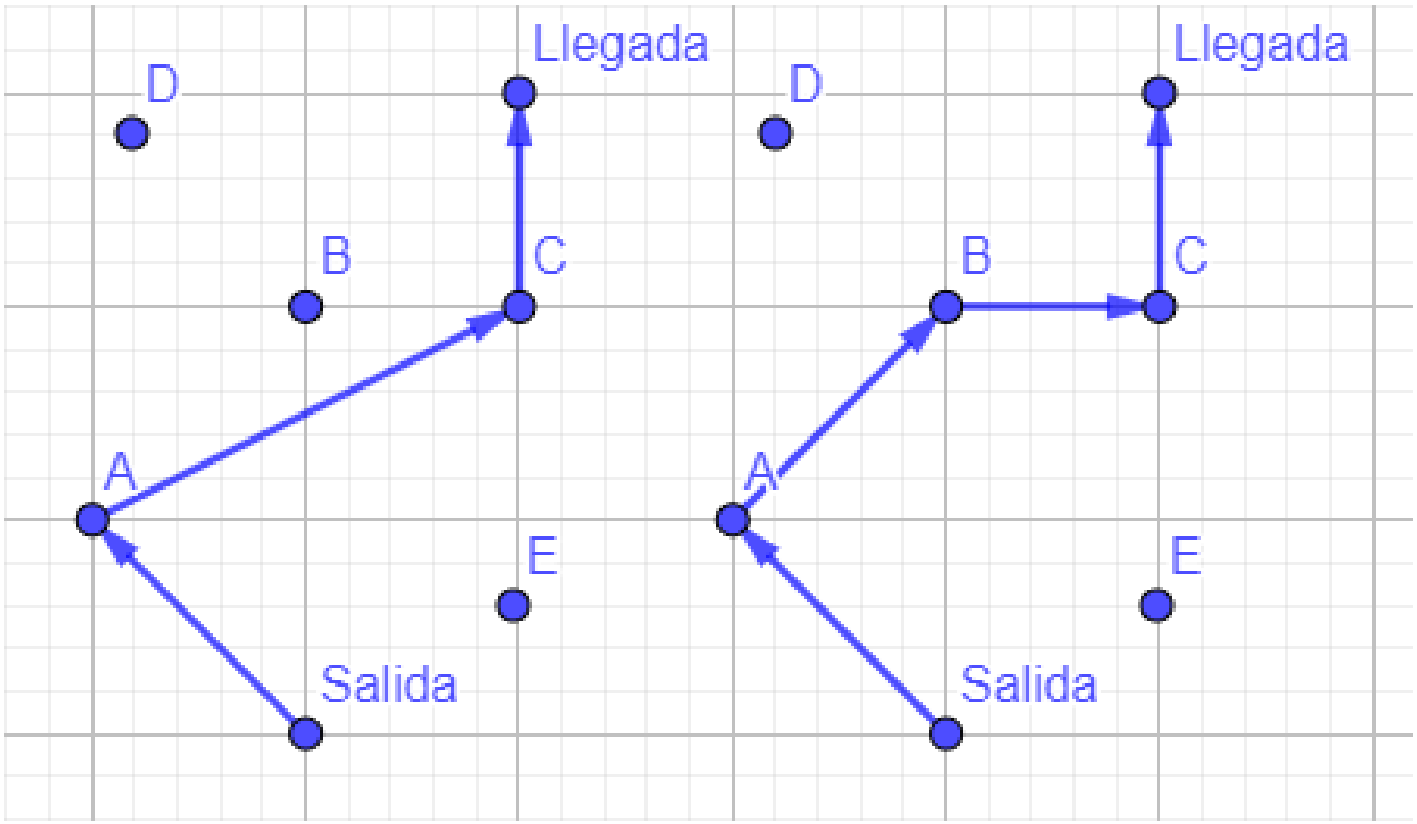
Búsqueda local: 2-Opt

- Dado un vehículo, cambia el orden de los clientes visitados con el objetivo de reducir la distancia recorrida.



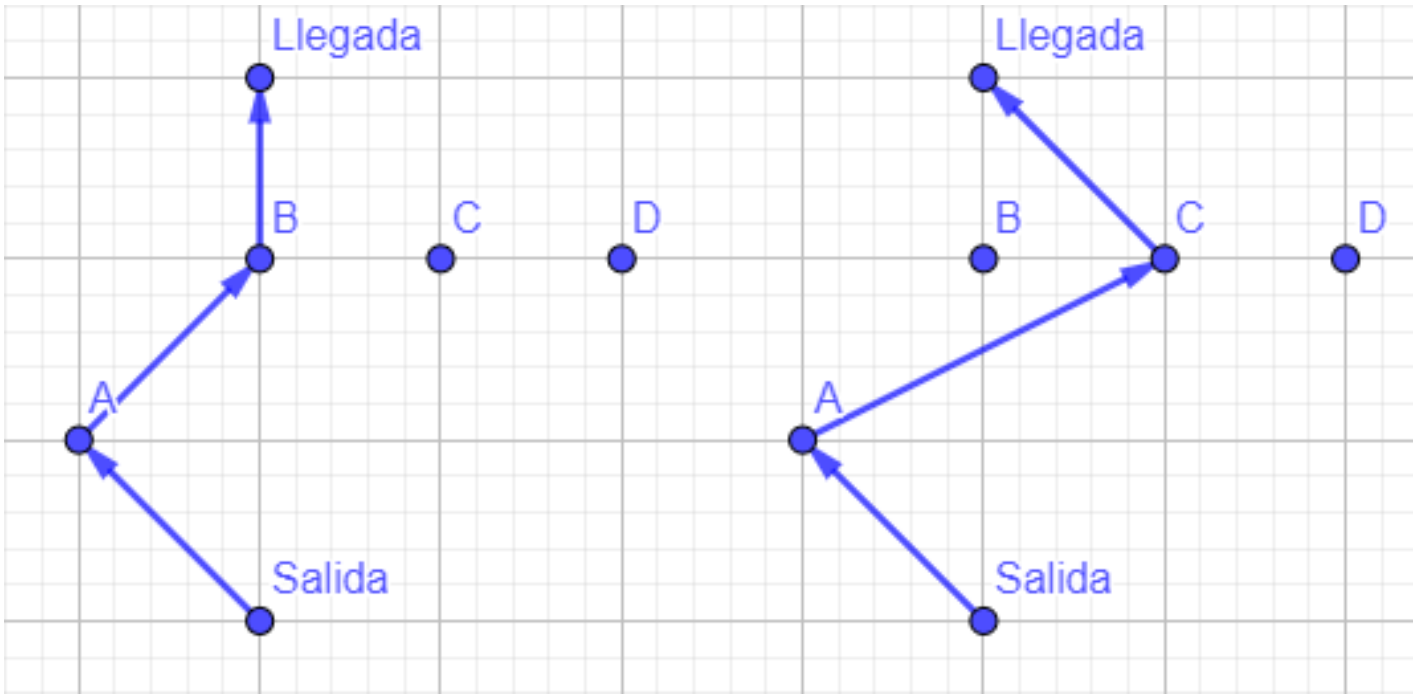
Búsqueda local: Insert

- Intenta insertar un cliente no visitado en la ruta de un vehículo.



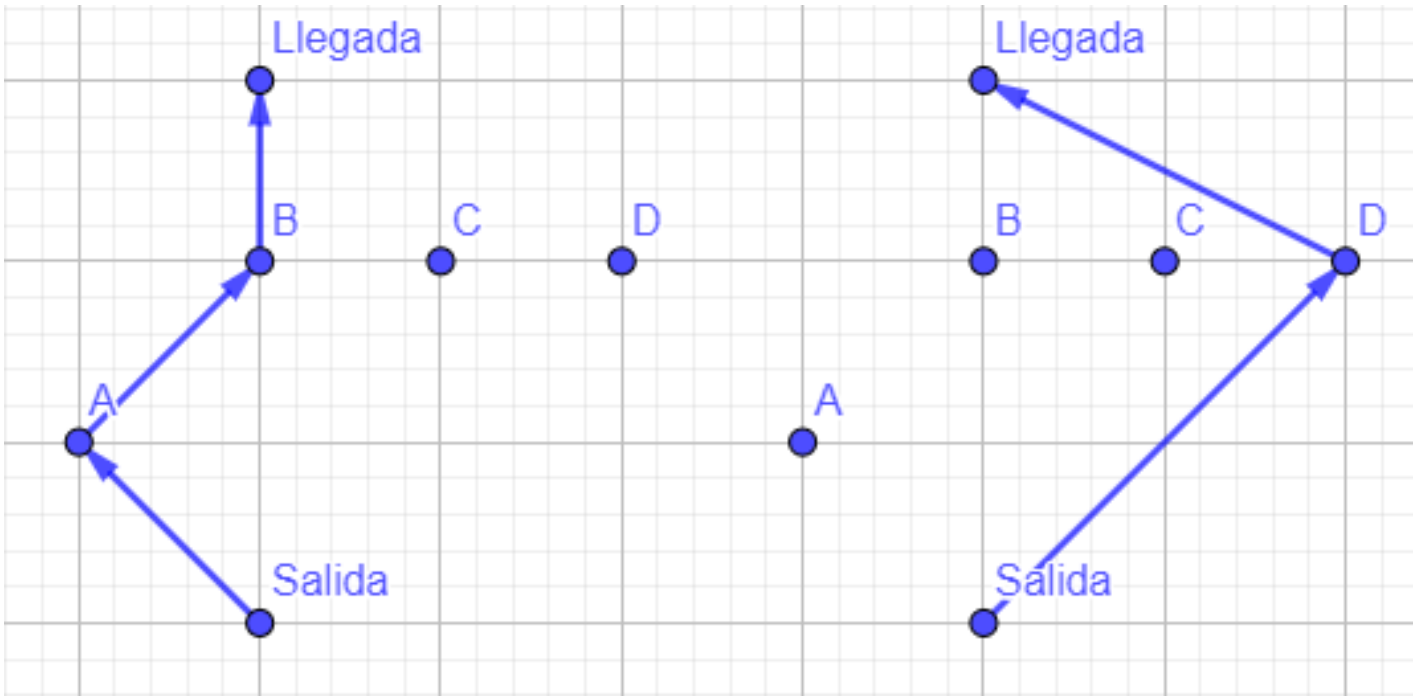
Búsqueda local: Replace Simple

- Dado un vehículo, intenta reemplazar un cliente visitado por un cliente no visitado en su ruta.



Búsqueda local: Replace Multiple

- Dado un vehículo, intenta reemplazar uno o varios clientes visitados por un cliente no visitado en su ruta.



Centro de gravedad (COG)

- En las búsquedas locales que buscan clientes candidatos a agregar entre los no visitados, se priorizan aquellos más cercanos al centro de gravedad de la ruta.

Centro de gravedad (COG)

- En las búsquedas locales que buscan clientes candidatos a agregar entre los no visitados, se priorizan aquellos más cercanos al centro de gravedad de la ruta.
- El COG es un punto en el plano, por lo tanto, tiene una coordenada x y una coordenada y.

Centro de gravedad (COG)

- En las búsquedas locales que buscan clientes candidatos a agregar entre los no visitados, se priorizan aquellos más cercanos al centro de gravedad de la ruta.
- El COG es un punto en el plano, por lo tanto, tiene una coordenada x y una coordenada y.
- La coordenada x_{COG} es el promedio de todas las coordenadas x de los clientes de la ruta ponderados por el beneficio del cliente. (Idem y_{COG}).

Centro de gravedad (COG)

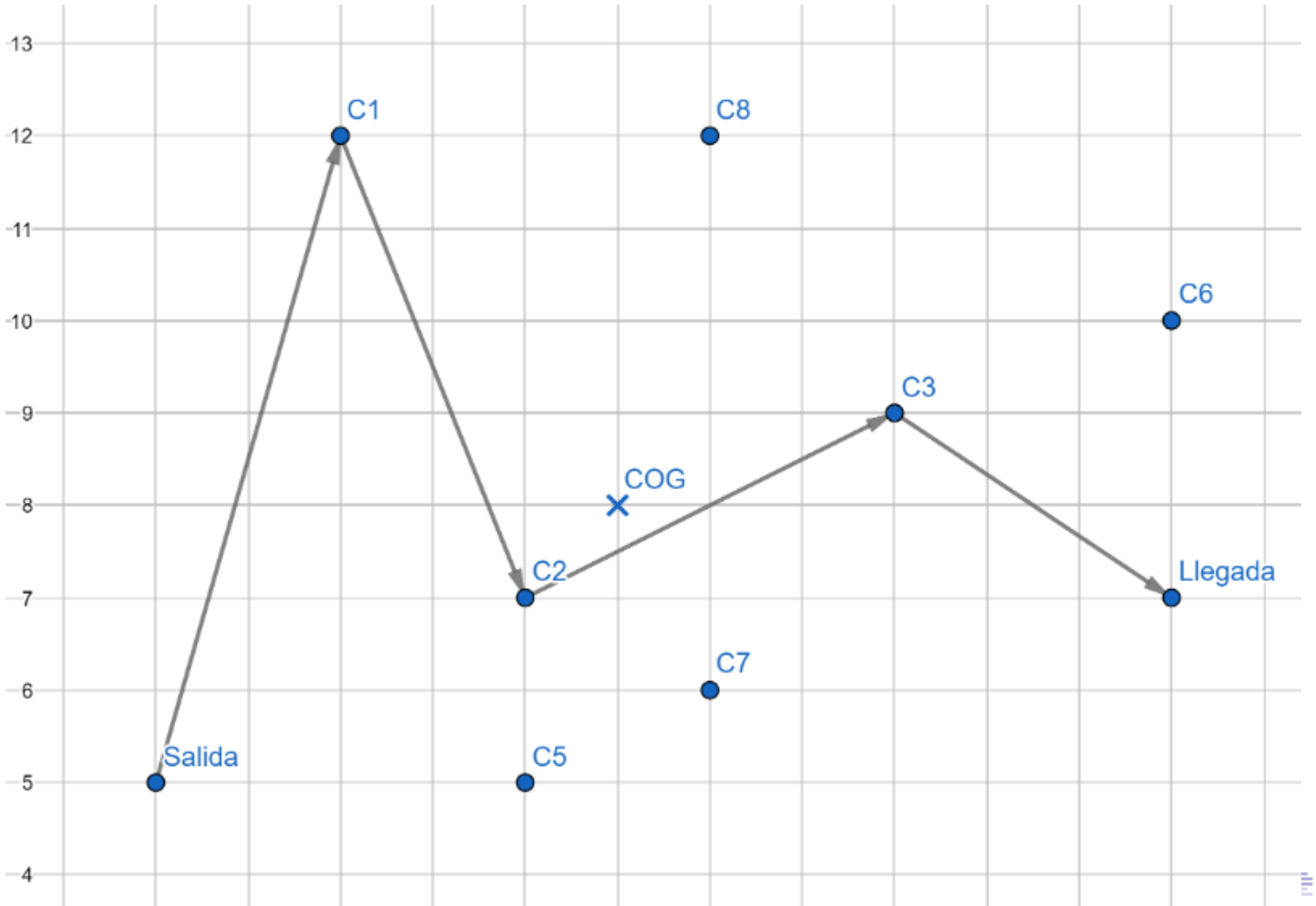
- En las búsquedas locales que buscan clientes candidatos a agregar entre los no visitados, se priorizan aquellos más cercanos al centro de gravedad de la ruta.
- El COG es un punto en el plano, por lo tanto, tiene una coordenada x y una coordenada y.
- La coordenada x_{COG} es el promedio de todas las coordenadas x de los clientes de la ruta ponderados por el beneficio del cliente. (Idem y_{COG}).
- $x_{cog} = (\sum_{\forall i \in ruta} x_i * B_i) / \sum_{\forall i \in ruta} B_i$

Centro de gravedad (COG)

- En las búsquedas locales que buscan clientes candidatos a agregar entre los no visitados, se priorizan aquellos más cercanos al centro de gravedad de la ruta.
- El COG es un punto en el plano, por lo tanto, tiene una coordenada x y una coordenada y.
- La coordenada x_{COG} es el promedio de todas las coordenadas x de los clientes de la ruta ponderados por el beneficio del cliente. (Idem y_{COG}).
- $x_{cog} = (\sum_{\forall i \in ruta} x_i * B_i) / \sum_{\forall i \in ruta} B_i$
- $y_{cog} = (\sum_{\forall i \in ruta} y_i * B_i) / \sum_{\forall i \in ruta} B_i$

Centro de gravedad (COG)

- La figura muestra el centro de gravedad asumiendo que todos los clientes en la ruta tienen el mismo beneficio. Orden de los clientes según su cercanía al COG: c_7, c_5, c_8 y c_6.



Codificación de las soluciones mejoradas

- Una vez que se modifican las soluciones, debemos actualizar su codificación de modo tal que al decodificarla obtengamos la solución mejorada en vez de su versión anterior.

Codificación de las soluciones mejoradas

- Una vez que se modifican las soluciones, debemos actualizar su codificación de modo tal que al decodificarla obtengamos la solución mejorada en vez de su versión anterior.
- Si no actualizamos su codificación (vector de enteros aleatorios), cuando se utilice la solución mejorada como padre en el *crossover* sus características viejas serán las que heredará la solución resultante.

Codificación de las soluciones mejoradas

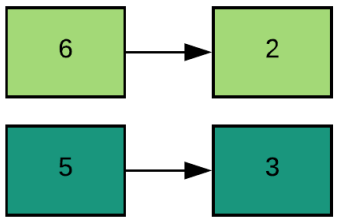
- Una vez que se modifican las soluciones, debemos actualizar su codificación de modo tal que al decodificarla obtengamos la solución mejorada en vez de su versión anterior.
- Si no actualizamos su codificación (vector de enteros aleatorios), cuando se utilice la solución mejorada como padre en el *crossover* sus características viejas serán las que heredará la solución resultante.

Dado el siguiente vector ordenado de RandomKeys:

Key	7	13	21	27	45	54	79	89
ClientId	6	2	5	1	4	8	3	7

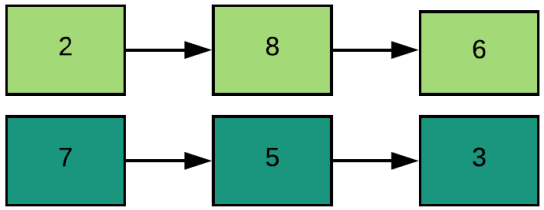
Hash de la solución generada: 6@2#5@3

El decodificar goloso genera una solución que contiene las rutas:



Codificación de las soluciones mejoradas

Las búsquedas locales mejoran las rutas agregando clientes y modificando el orden del recorrido:



El encoder actualiza el mapeo entre Key y ClientId del vector de RandomKeys:

Key	7	13	21	27	45	54	79	89
ClientId	2	8	6	7	5	3	1	4

Hash de la nueva solución: 2@8@6#7@5@3

- En la figura podemos ver que no se modificaron los *keys* ni los *ClientId*, lo que se modificó fueron las asociaciones entre ellos. De forma tal que cuando se ordena el vector de enteros aleatorios según los *keys*, obtenemos los clientes ordenados por vehículo y luego el orden en que serán visitados.

Orden en que se aplican las búsquedas locales

- En cada nueva generación se le aplican las búsquedas locales a la mejor solución (que no haya sido mejorada en una generación anterior).

Orden en que se aplican las búsquedas locales

- En cada nueva generación se le aplican las búsquedas locales a la mejor solución (que no haya sido mejorada en una generación anterior).
- El orden en que se aplican las estrategias de búsqueda local es importante.

Orden en que se aplican las búsquedas locales

- En cada nueva generación se le aplican las búsquedas locales a la mejor solución (que no haya sido mejorada en una generación anterior).
- El orden en que se aplican las estrategias de búsqueda local es importante.
- Por ejemplo, el *2-Opt* busca reducir la distancia recorrida de un vehículo. Por lo tanto, si aplicamos el *2-Opt* al final, no hacemos uso de la distancia ahorrada y la aptitud de la solución resultante no incrementa.

Orden en que se aplican las búsquedas locales

- En cada nueva generación se le aplican las búsquedas locales a la mejor solución (que no haya sido mejorada en una generación anterior).
- El orden en que se aplican las estrategias de búsqueda local es importante.
- Por ejemplo, el *2-Opt* busca reducir la distancia recorrida de un vehículo. Por lo tanto, si aplicamos el *2-Opt* al final, no hacemos uso de la distancia ahorrada y la aptitud de la solución resultante no incrementa.
- Realicé varias pruebas modificando el orden en que se aplican las búsquedas locales, medí sus tiempos y los resultados finales.

Orden en que se aplican las búsquedas locales

- En cada nueva generación se le aplican las búsquedas locales a la mejor solución (que no haya sido mejorada en una generación anterior).
- El orden en que se aplican las estrategias de búsqueda local es importante.
- Por ejemplo, el *2-Opt* busca reducir la distancia recorrida de un vehículo. Por lo tanto, si aplicamos el *2-Opt* al final, no hacemos uso de la distancia ahorrada y la aptitud de la solución resultante no incrementa.
- Realicé varias pruebas modificando el orden en que se aplican las búsquedas locales, medí sus tiempos y los resultados finales.
- Para tales pruebas mantuve constante el resto de la configuración del BRKGA de modo que no influya en el resultado.

Orden en que se aplican las búsquedas locales

- Siglas de las búsquedas locales.

Orden en que se aplican las búsquedas locales

- Siglas de las búsquedas locales.
- Swap: **S**

Orden en que se aplican las búsquedas locales

- Siglas de las búsquedas locales.
- Swap: **S**
- 2-Opt: **O**

Orden en que se aplican las búsquedas locales

- Siglas de las búsquedas locales.
- Swap: **S**
- 2-Opt: **O**
- Insert: **I**

Orden en que se aplican las búsquedas locales

- Siglas de las búsquedas locales.
- Swap: **S**
- 2-Opt: **O**
- Insert: **I**
- Replace Simple: **Rs**

Orden en que se aplican las búsquedas locales

- Siglas de las búsquedas locales.
- Swap: **S**
- 2-Opt: **O**
- Insert: **I**
- Replace Simple: **Rs**
- Replace Múltiple: **Rm**

Orden en que se aplican las búsquedas locales

- Siglas de las búsquedas locales.
- Swap: **S**
- 2-Opt: **O**
- Insert: **I**
- Replace Simple: **Rs**
- Replace Múltiple: **Rm**
- Por lo tanto, si en la fila dice **SRsOIRm** significa que las búsquedas aplicaron en el siguiente orden: Swap, Replace Simple, 2-Opt, Insert y Raplace Multiple.

Orden en que se aplican las búsquedas locales

- Resultados de 25 ejecuciones sobre la instancia *p5.3.x*. En la siguiente tabla se ven 7 posibles combinaciones de las búsquedas.

Orden BL	T_{avg}	B_{min}	B_{avg}	B_{max}	i_{eAvg}	i_{eMax}	*Best*
IRmRsOS	49397	1460	1485	1540	0.95	0.99	1555
ORsSIRm	39576	1485	1509	1525	0.97	0.98	1555
SIORsSORm	43556	1495	1512	1535	0.97	0.99	1555
SOIORsRmSORm	49449	1505	1522	1545	0.98	0.99	1555
SOIRsRm	36595	1500	1512	1525	0.97	0.98	1555
SOSIRsSORm	40375	1505	1521	1535	0.98	0.99	1555
SRsOIRm	43423	1480	1510	1535	0.97	0.99	1555

Orden en que se aplican las búsquedas locales

- Resultados de 25 ejecuciones sobre la instancia *p7.4.t*. En la siguiente tabla se ven 7 posibles combinaciones de las búsquedas.

Orden BL	T_{avg}	B_{min}	B_{avg}	B_{max}	i_{eAvg}	i_{eMax}	*Best*
IRmRsOS	81876	1004	1038	1064	0.96	0.99	1077
ORsSIRm	86537	1024	1038	1063	0.96	0.99	1077
SIORsSORm	91839	1033	1049	1077	0.97	1.00	1077
SOIORsRmSORm	126705	1042	1055	1069	0.98	0.99	1077
SOIRsRm	82889	1032	1047	1071	0.97	0.99	1077
SOSIRsSORm	94731	1038	1055	1071	0.98	0.99	1077
SRsOIRm	90306	1024	1042	1067	0.97	0.99	1077

Resultados

Se compararon los resultados con los de los siguientes trabajos:

- *The team orienteering problem*. Implementaron una optimización multinivel. (CGW) Chao I-M., Golden B.L. yWasil E.A. European Journal of Operational Research, 88:464-474, (1996).

Resultados

Se compararon los resultados con los de los siguientes trabajos:

- *The team orienteering problem*. Implementaron una optimización multinivel. (CGW) Chao I-M., Golden B.L. yWasil E.A. European Journal of Operational Research, 88:464-474, (1996).
- *A tabu search heuristic for the team orienteering problem*. (TMH) Tang H. y Miller-Hooks E. Computers and Operations Research, 32:1379-1407, (2005).

Resultados

- *Metaheuristics for the team orienteering problem*. Propusieron 2 Variable Neighborhood Search y 2 Búsquedas Tabú, los mejores resultados los obtuvieron en su VNS slow. (VNS_{slow}) Archetti C., Hertz A. y M.G. Speranza. Journal of Heuristics, 13:49-76, (2007).

Resultados

- *Metaheuristics for the team orienteering problem*. Propusieron 2 Variable Neighborhood Search y 2 Búsquedas Tabú, los mejores resultados los obtuvieron en su VNS slow. (VNS_{slow}) Archetti C., Hertz A. y M.G. Speranza. Journal of Heuristics, 13:49-76, (2007).
- *Ants can solve the team orienteering problem*. (ACO_{seq}) Ke L., Archetti C. y Feng Z. Computers and Industrial Engineering, 54:648-665, (2008).

Resultados

- *Metaheuristics for the team orienteering problem*. Propusieron 2 Variable Neighborhood Search y 2 Búsquedas Tabú, los mejores resultados los obtuvieron en su VNS slow. (VNS_{slow}) Archetti C., Hertz A. y M.G. Speranza. Journal of Heuristics, 13:49-76, (2007).
- *Ants can solve the team orienteering problem*. (ACO_{seq}) Ke L., Archetti C. y Feng Z. Computers and Industrial Engineering, 54:648-665, (2008).
- *A memetic algorithm for the team orienteering problem*. (MA) Bouly H., Dang D.-C. y Moukrim A. A Quarterly Journal of Operations Research, 8:49-70, (2010).

Resultados

- Se ejecutó 3 veces la versión final sobre cada una de las 345 instancias del benchmarck.

Resultados

- Se ejecutó 3 veces la versión final sobre cada una de las 345 instancias del benchmarck.
- La tabla muestra el resultado para 6 instancias representativas del benchmark.

Instancia	N/V/D	B_{min}	B_{avg}	B_{max}	i_{eAvg}	i_{eMax}	*Best*
p2.2.k	21/2/22.50	270	274	275	1.00	1.00	275
p2.3.g	21/3/10.70	145	145	145	1.00	1.00	145
p3.4.p	33/4/22.50	560	560	560	1.00	1.00	560
p5.3.x	66/3/40.00	1515	1524	1550	0.98	1.00	1555
p7.2.e	102/2/50.00	290	290	290	1.00	1.00	290
p7.4.t	102/4/100.00	1047	1054	1064	0.98	0.99	1077

Resultados

- $\delta Z_{min} = \sum_{i \in instance} Best_i - B_{min}$

Resultados

- $\delta Z_{min} = \sum_{i \in instance} Best_i - B_{min}$
- $\delta Z_{max} = \sum_{i \in instance} Best_i - B_{max}$

Resultados

- $\delta Z_{min} = \sum_{i \in instance} Best_i - B_{min}$
- $\delta Z_{max} = \sum_{i \in instance} Best_i - B_{max}$
- $\delta Z = \delta Z_{min} - \delta Z_{max}$

Resultados

- $\delta Z_{min} = \sum_{i \in instance} Best_i - B_{min}$
- $\delta Z_{max} = \sum_{i \in instance} Best_i - B_{max}$
- $\delta Z = \delta Z_{min} - \delta Z_{max}$

Algoritmo	δZ_{min}	δZ_{max}	δZ
CGW	4340	-	-
$BRKGA_{bl}$	3054	1636	1418
TMH	2404	-	-
$TS_{penalty}$	2376	981	1395
$TS_{feasible}$	1184	399	785
VNS_{fast}	1436	352	1084
VNS_{slow}	427	84	343
ACO_{seq}	-	204	-
MA	434	80	354

Conclusiones

- Un 70% de los resultados llegaron a la mejor solución conocida de la instancia testeada.

Conclusiones

- Un 70% de los resultados llegaron a la mejor solución conocida de la instancia testeada.
- El restante 30% obtuvo valores competitivos con los mejores trabajos previos.

Conclusiones

- Un 70% de los resultados llegaron a la mejor solución conocida de la instancia testeada.
- El restante 30% obtuvo valores competitivos con los mejores trabajos previos.
- El BRKGA sin búsqueda local no dió buenos resultados para instancias grandes del problema.

Conclusiones

- Un 70% de los resultados llegaron a la mejor solución conocida de la instancia testeada.
- El restante 30% obtuvo valores competitivos con los mejores trabajos previos.
- El BRKGA sin búsqueda local no dió buenos resultados para instancias grandes del problema.
- Se implementaron variantes de decodificadores.

Conclusiones

- Un 70% de los resultados llegaron a la mejor solución conocida de la instancia testeada.
- El restante 30% obtuvo valores competitivos con los mejores trabajos previos.
- El BRKGA sin búsqueda local no dió buenos resultados para instancias grandes del problema.
- Se implementaron variantes de decodificadores.
- Se implementaron dos formas de calcular el hash para resolver el problema de individuos repetidos.

Conclusiones

- Un 70% de los resultados llegaron a la mejor solución conocida de la instancia testeada.
- El restante 30% obtuvo valores competitivos con los mejores trabajos previos.
- El BRKGA sin búsqueda local no dió buenos resultados para instancias grandes del problema.
- Se implementaron variantes de decodificadores.
- Se implementaron dos formas de calcular el hash para resolver el problema de individuos repetidos.
- Se implementaron diversas búsquedas locales y un codificador de soluciones para mantener la consistencia entre los individuos y sus genes.

Conclusiones

- Un 70% de los resultados llegaron a la mejor solución conocida de la instancia testeada.
- El restante 30% obtuvo valores competitivos con los mejores trabajos previos.
- El BRKGA sin búsqueda local no dió buenos resultados para instancias grandes del problema.
- Se implementaron variantes de decodificadores.
- Se implementaron dos formas de calcular el hash para resolver el problema de individuos repetidos.
- Se implementaron diversas búsquedas locales y un codificador de soluciones para mantener la consistencia entre los individuos y sus genes.
- Se realizaron múltiples pruebas de eficiencia variando las configuraciones generales.

Trabajos Futuros

- Sería útil una herramienta para visualizar los caminos generados.

Trabajos Futuros

- Sería útil una herramienta para visualizar los caminos generados.
- Investigar otras variantes de decodificadores. Ej, particionar los clientes según su centro de gravedad, asignar un vehículo a cada centro y seleccionar un orden de clientes por centro de gravedad según un vector de enteros aleatorios.

Trabajos Futuros

- Sería útil una herramienta para visualizar los caminos generados.
- Investigar otras variantes de decodificadores. Ej, particionar los clientes según su centro de gravedad, asignar un vehículo a cada centro y seleccionar un orden de clientes por centro de gravedad según un vector de enteros aleatorios.
- Investigar otros métodos de *crossover*. Ej, Que cada alelo represente un vehículo con su ruta en vez de un cliente.

Trabajos Futuros

- Sería útil una herramienta para visualizar los caminos generados.
- Investigar otras variantes de decodificadores. Ej, particionar los clientes según su centro de gravedad, asignar un vehículo a cada centro y seleccionar un orden de clientes por centro de gravedad según un vector de enteros aleatorios.
- Investigar otros métodos de *crossover*. Ej, Que cada alelo represente un vehículo con su ruta en vez de un cliente.
- Analizar si existe alguna otra variante del BRKGA que genere buenos resultados sin depender de las búsquedas locales.

Gracias!