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

Alejandro Lix Klett
Directora: Irene Loiseau

Contenido

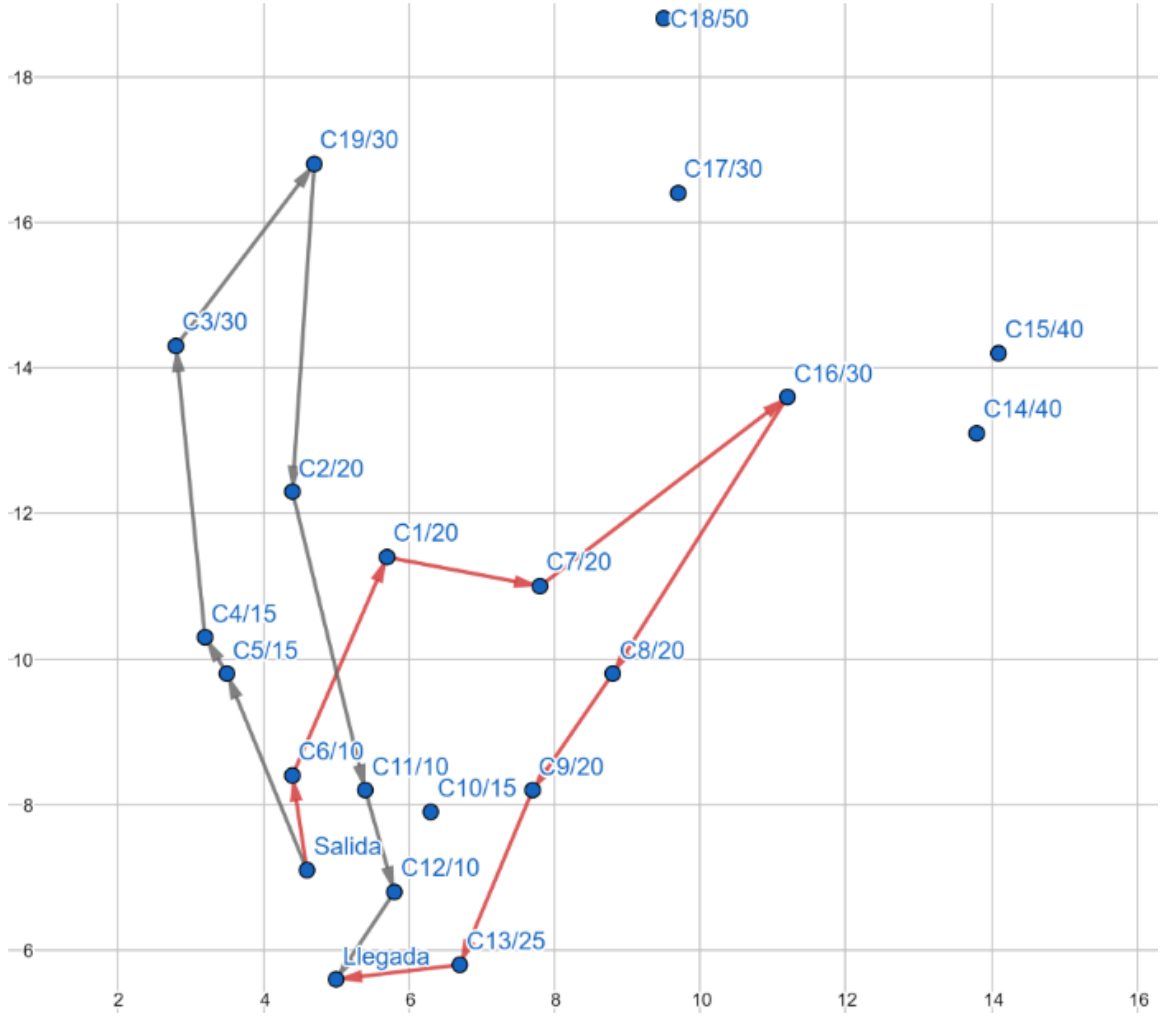
- Team Orienteering Problem
  - Origen
  - Descripción
  - Función objetivo
- Biased Random Key Genetic Algorithm
  - Algoritmos genéticos
  - RKGA
  - BRKGA
- Algoritmo Propuesto
  - BRKGA
    - Decodificadores
    - Configuración general
  - Búsquedas Locales
- Resultados
  - Benchmarck de instancias
  - Trabajos Previos
  - Resultados Particulares
  - Resultados Globales
- Concluciones
- Trabajos Futuros

- Orientación, 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 limite preespecificado. Cada punto de control tiene un puntaje. El objetivo es maximizar el puntaje total.
  - Este problema se conoce como Orienteering Problem (OP). El OP es NP-Completo como demostraron Golden, Levy y Vohra.
- Team Orienteering Problem (TOP)
  - Extiende a OP, como lo contiene, es al menos tan dificil. Hay multiples vehiculos y no deben visitar los mismos clientes ya que su beneficio solo puede ser recolectado una sola vez.

- Definición informal:
  - Existen N vehículos
  - Todos los vehículos salen de un punto de salida y deben llegar al punto de llegada habiendo recorrido una distancia menor a dmax
  - Existen M clientes, cada uno tiene un beneficio bi y una coordena en el plano
  - Los puntos de salida y llegada tienen un beneficio de cero
  - El beneficio de los clientes solo puede ser recolectado una vez
  - Se utiliza la distancia euclidiana
- Función Objetivo
  - Maximizar la sumatoria de los beneficios recolectados de todos los vehículos

Team Orienteering Problem

Instancia p2.2.k del benchmark, author Tsiligirides. Tiene dos vehículos con un dmax = 22,50. Una posible solución optima para el problema:

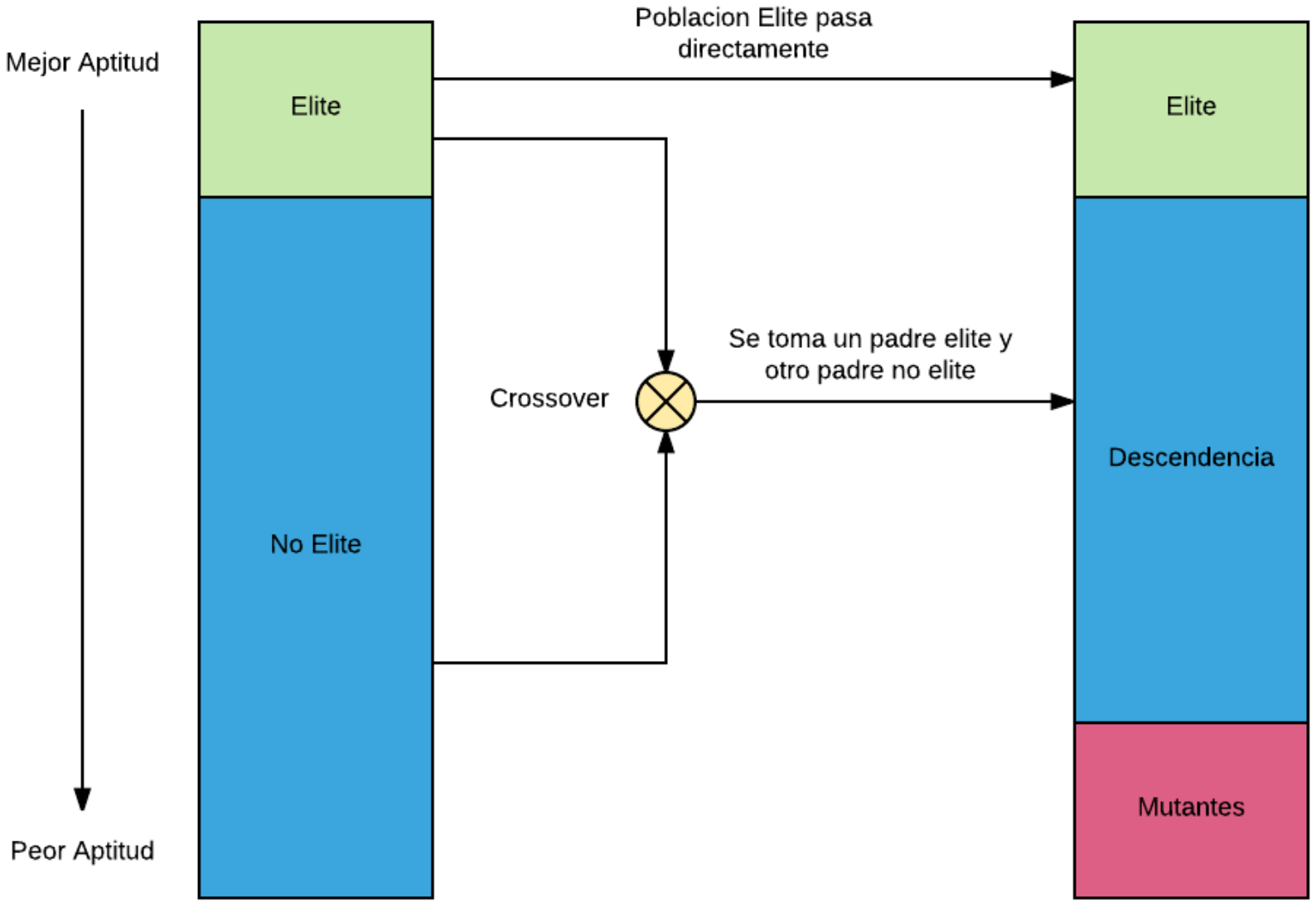


- Algoritmos Genéticos (GA)
  - Motivados en el concepto de supervivencia del más apto
  - Los algoritmos genéticos manejan un conjunto de individuos
  - Cada individuo es un cromosoma que codifica una solución
  - Cada cromosoma tienen asociado un nivel de condición física que está correlacionado con 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.

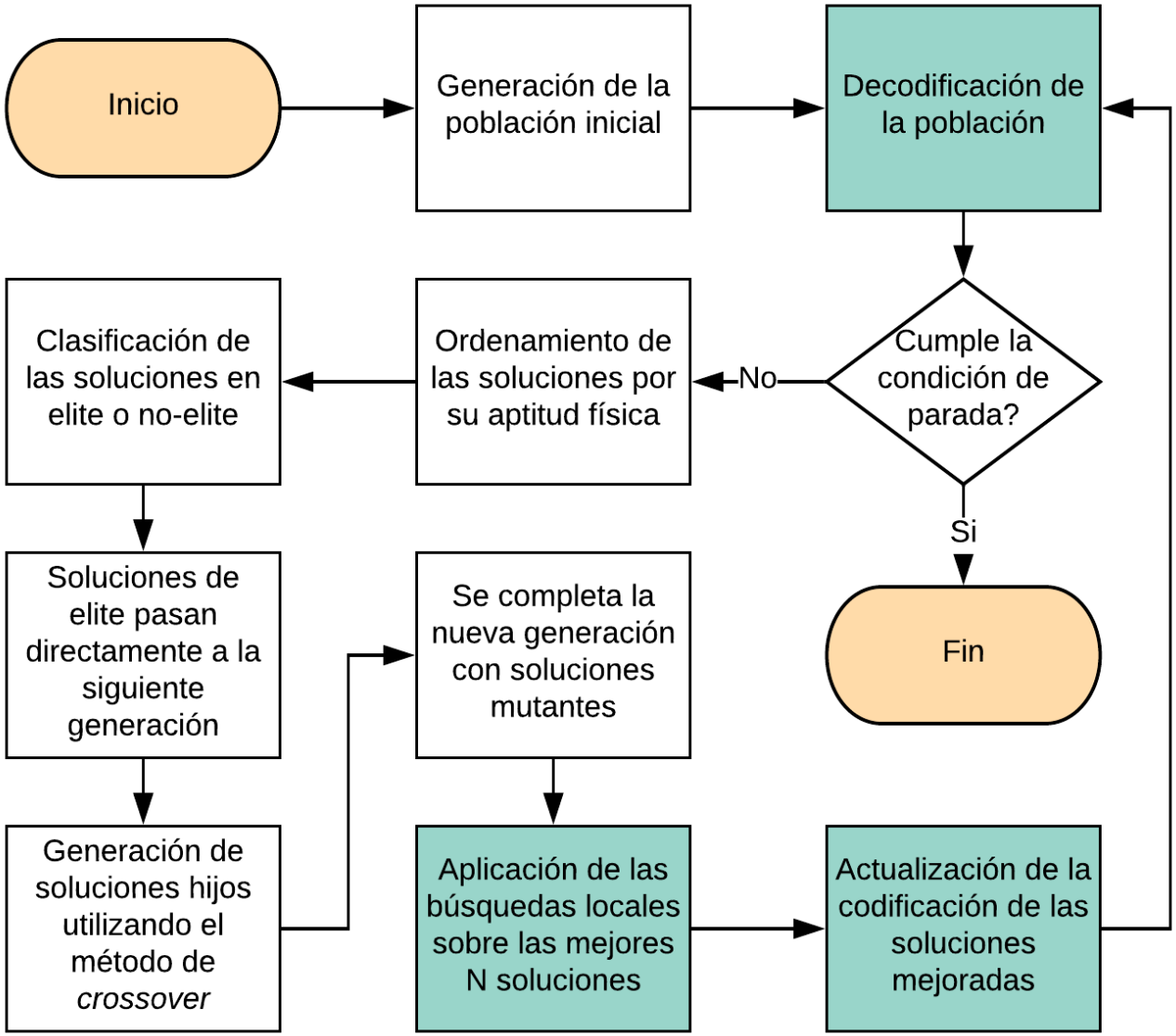
- Random Key Genetic Algorithm (RKGA)
  - Los individuos son representados por un vector de números reales en el intervalo [0, 1]
  - La poblacion inicial es generada al azar
  - El decodificador es el responsable de convertir un cromosoma en una solución válida del problema
  - En cada iteración se toman los mejores individuos y pasan directamente a la siguiente generación (elites)
  - La mayoria de los individuos de la nueva generacion se generan crusando dos individuos de la generacion actual (crossover)
  - Por último un porcentaje muy bajo son generados 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.
  - Parameterized Uniform Crossover. La probabilidad de que se trasmita el alelo del padre de elite es mayor que la del padre de no-elte.

Biased Random Key Genetic Algorithm



Algoritmo Propuesto



- Generación de la población inicial
  - Se crea una cantidad de vectores de enteros aleatorios igual a la cantidad de soluciones por generacion que se desea
- Decodificación de la población
  - Orden en que se seleccionan los clientes
  - Decodificador Simple
  - Decodificador Goloso
  - Cada vector de enteros se decodifica en una solución válida del problema

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

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

- Evolución de población
  - Ordenamiento de las soluciones por aptitud física
  - Clasificación de las soluciones en elite o no-elite
  - Soluciones de elite pasan directamente a la siguiente generación
  - Generación de soluciones hijos utilizando el método de *crossover*
  - No se permiten soluciones repetidas, se utiliza el hash para determinar si dos soluciones son iguales
  - Se completa la nueva generación con soluciones mutantes, soluciones aleatorias para escapar de los mínimos locales

- Evaluación de la condición de parada
  - Mínima cantidad de generaciones
  - Ultimas X generaciónes sin que haya mejorado el beneficio de la mejor solución
  - Si no se cumplen las condiciones, comienza un nuevo ciclo evolucionando la población
  - Si se cumplen, retorna la mejor solución

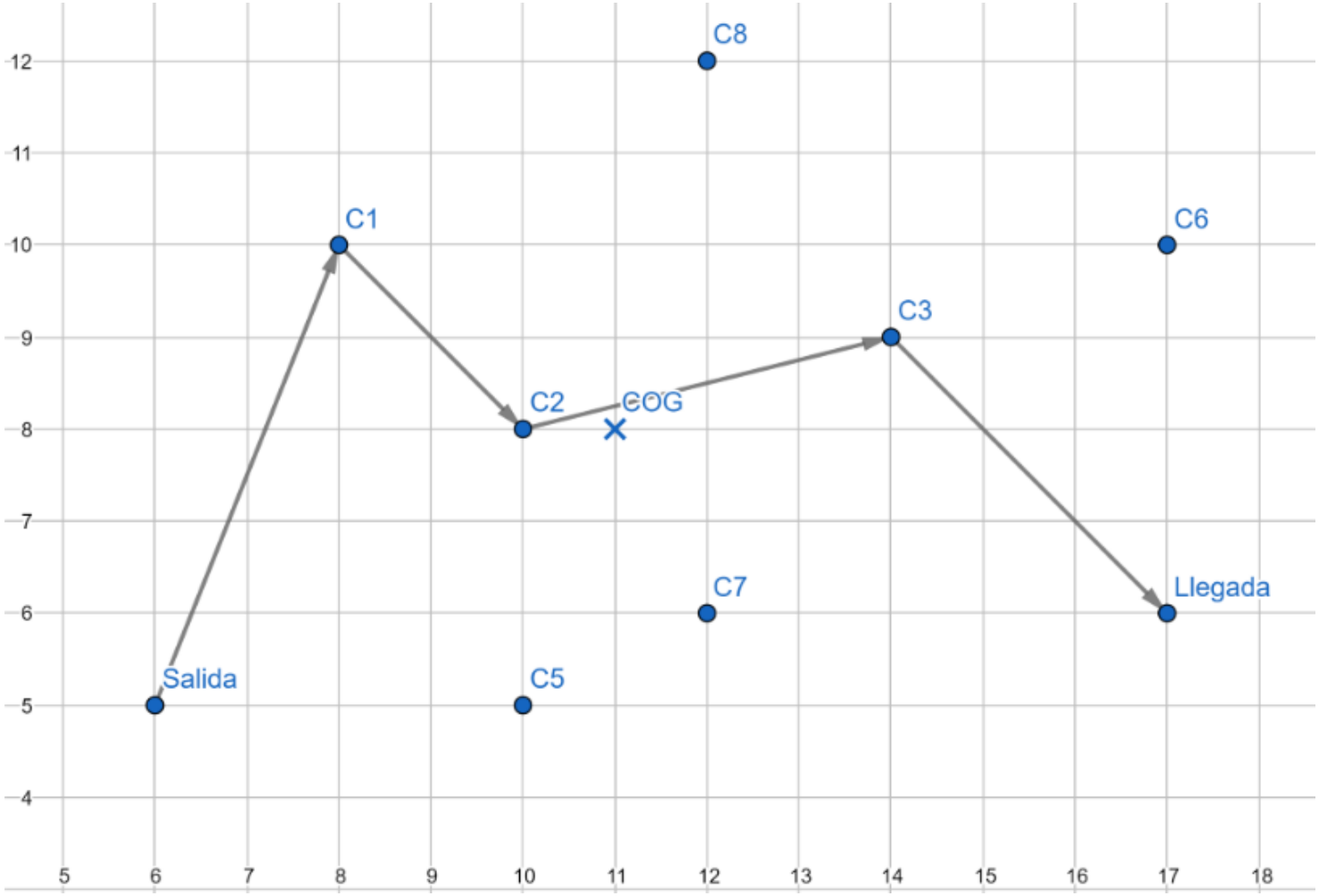
- Hash de un individuo
  - Se evaluaron dos opciones
  - Opción 1: 6@2@5@1@4@8@3@7
  - Opción 2: 5@1@3#6@2@8@

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

- Búsquedas locales a las mejores N soluciones
  - Se aplica una secuencia de BL a las mejores N soluciones en cada generacion
  - **Swap**: Intercambio de clientes entre distintos vehículos
  - **2-Opt**: Reordenamiento de clientes a visitar para un vehículo
  - **Insert**: Insertar cliente no visitado en alguna ruta
  - **Replace Simple**: Insertar un cliente no visitado por uno visitado en alguna ruta
  - **Replace Multiple**: Insertar un cliente no visitado por uno a varios en alguna ruta
  - Se activa el cálculo del COG para Insert y Replace. A cada cambio en la ruta del vehículo el COG se actualiza

- Centro de Gravedad (COG)
  - Orden respecto del COG: C7, C5, C8 y C6



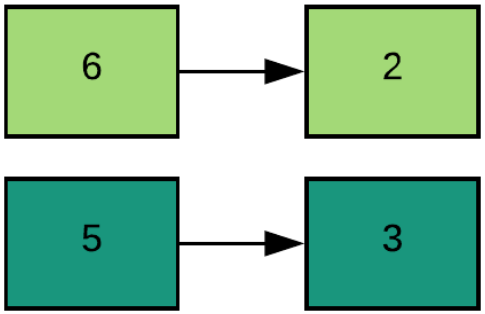
- Codificación de las soluciones mejoradas
  - Actualización del código genético post búsqueda local

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

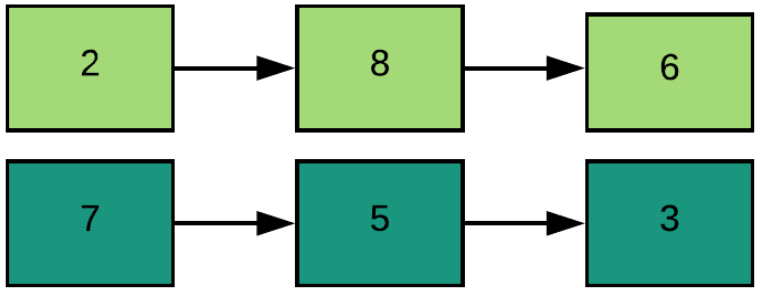
Hash de la solución generada: 6@2#5@3

El decodificar goloso genera una solución que contiene las rutas:



Algoritmo Propuesto

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

- Se compararon los resultados con los siguientes trabajos previos:
  - Optimización multinivel de Chao et al. (CGW)
  - Tabu Search de Tang y Miller-Hooks (TMH)
  - Memetic Algorithm (MA) de Bouly et al.
  - Ant Colony Optimization (ACOseq) de Ke et al.
  - Variable Neighborhood Search (VNSslow) de Archetti et al.

- Resultados brkga puro decodificador simple y goloso:

Instancia	N/V/D	B_{min}	B_{avg}	B_{max}	ACOseq	VNSslow	MA	i_{eAvg}	i_{eMax}	$Best$
p2.2.k	21/2/22.50	230	231	240	275	275	275	0.84	0.87	275
p2.3.g	21/3/10.70	140	142	145	145	145	145	0.98	1.00	145
p3.4.p	33/4/22.50	320	329	340	560	560	560	0.59	0.61	560
p5.3.x	66/3/40.00	680	680	680	1540	1555	1555	0.44	0.44	1555
p7.2.e	102/2/50.00	123	138	157	290	290	290	0.48	0.54	290
p7.4.t	102/4/100.00	260	285	310	1077	1077	1077	0.26	0.29	1077

Instancia	N/V/D	B_{min}	B_{avg}	B_{max}	ACOseq	VNSslow	MA	i_{eAvg}	i_{eMax}	$Best$
p2.2.k	21/2/22.50	240	253	260	275	275	275	0.92	0.95	275
p2.3.g	21/3/10.70	145	145	145	145	145	145	1.00	1.00	145
p3.4.p	33/4/22.50	420	435	450	560	560	560	0.78	0.80	560
p5.3.x	66/3/40.00	735	735	735	1540	1555	1555	0.47	0.47	1555
p7.2.e	102/2/50.00	200	209	222	290	290	290	0.72	0.77	290
p7.4.t	102/4/100.00	461	478	505	1077	1077	1077	0.44	0.47	1077

- Resultados brkga con las búsquedas locales:

Instancia	N/V/D	B_{min}	B_{avg}	B_{max}	ACOseq	VNSslow	MA	i_{eAvg}	i_{eMax}	$Best$
p2.2.k	21/2/22.50	270	274	275	275	275	275	1.00	1.00	275
p2.3.g	21/3/10.70	145	145	145	145	145	145	1.00	1.00	145
p3.4.p	33/4/22.50	560	560	560	560	560	560	1.00	1.00	560
p5.3.x	66/3/40.00	1515	1524	1550	1540	1555	1555	0.98	1.00	1555
p7.2.e	102/2/50.00	290	290	290	290	290	290	1.00	1.00	290
p7.4.t	102/4/100.00	1047	1054	1064	1077	1077	1077	0.98	0.99	1077

- Síntesis resultados:

Algoritmo	δZ_{min}	δZ_{max}	δZ
CGW	4340	-	-
BRKGA	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

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

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

$$\delta Z = \delta Z_{\max} - \delta Z_{\min}$$

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 puro no es bueno para instancias grandes del problema.

- Aportes a futuros trabajos con brkga:
  - 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
  - Todos los algoritmos fueron implementados teniendo en cuenta los ordenes de complejidad
  - Se realizaron multiples pruebas de eficiencia variando las configuraciones generales

- Seria útil una herramienta para visualizar los caminos generados.
- Investigar otras variantes de decodificadores
  - Particionar los clientes según su centro de gravedad, asigna vehículo a cada centro y asignar desde ahí
- Investigar otros métodos de crossover
  - Que cada alelo represente un vehículo con su ruta en vez de un cliente
- Analizar si existe alguna variante del brkga que genere buenos resultados sin depender de las búsquedas locales.
  - Si no existe se podría decir que el brkga no es bueno para TOP

Gracias!