Los mejores genes del CVRP

Nicolás Robayo Pardo[1]

[1] Universidad de los Andes
Cra 1 Nº 18A - 12, 111711, Bogotá, Colombia
n.robayo10@uniandes.edu.co

Resumen

El problema del CVRP es un problema NP-Hard presente en numerosas aplicaciones de logística y distribución de productos que son claves para la eficiencia de las empresas actuales. Esta tarea desarrolla un algoritmo genético que prueba dos métodos de crossover: uno inspirado en el VRP y otro típico de los algoritmos genéticos y prueba su eficacia. El algoritmo genético resulta ser una metaheuristica poderosa para resolver el VRP y requiere de una calibración adecuada de sus parámetros de métodos de mutación, crossover e inicialización.

1 Introducción

El VRP es el problema de optimización combinatoria más conocido debido a que es simple de entender, pero a la vez es muy difícil de resolver. Este consiste en determinar un set de rutas que visiten un grupo de clientes a mínimo costo. Se debe cumplir que todos los clientes deben ser atendidos y cada cliente solo puede ser atendido una única vez. El VRP tiene múltiples variantes que complejizan este problema y lo acercan a la vida real. Entre estas están el CVRP el cual involucra vehículos iguales con capacidad limitada, el TWVRP el cual involucra ventanas de tiempo en cada uno de los nodos y el PDP el cual tiene paquetes que recoger y dejar y reglas de precedencia entre los nodos.

El VRP es un problema muy importante hoy en día debido a la necesidad de las empresas de llevar servicios a los clientes a través de redes de distribución en sistemas logísticos. Debido a la competencia actual por satisfacer los requerimientos de los clientes, los negocios deben mejorar la eficiencia de sus redes de distribución para mejorar la competitividad y así afrontar una era de globalización económica y outsourcing [1].

El problema que atacaremos acá es el CVRP. El CVRP consiste en determinar el set de máximo m rutas a mínimo coste en donde cada ruta empieza y termina en el depot, donde cada cliente es visitado exactamente una vez por un vehículo, sujeto a la restricción de que el total de demanda atendida por cada vehículo no exceda la capacidad. Lawrence y Bruce [2] determinaron que este problema es no determinístico en tiempo polinomial, es decir NP-Hard y no hay algoritmos en tiempo pseudo-polinomial que puedan resolvero razonablemente por lo que el uso de metaheuristicas es lo más común en aplicaciones reales. Al usar métodos exactos solo se puede resolver a optimalidad instancias muy pequeñas debido a que el árbol de Branch and Bound crece exponencialmente con el número de nodos del problema. En entornos reales, las metaheuristicas son de mucha ayuda pues, a pesar de no garantizar optimalidad, obtienen muy buenas soluciones en un tiempo corto. Para el VRP se han usado varias metaheuristicas desde algoritmos genéticos, sistemas de colonia de hormigas hasta optimización de partículas.

Según Golderg (1989) [3] los atributos principales de los algoritmos Genéticos son:

(1) Los cálculos están basados en parámetros codificados, no en valores parametrizados
(2) Posee la capacidad de realizar búsquedas altamente paralelas, evitando caer en óptimos locales
(3) No tiene formulas matemáticamente complejas, solo la función de fitness debe ser calculada
(4) No tiene reglas especificas para guiar la dirección de búsqueda para un óptimo, se usa una regla aleatoria de búsqueda que usa la regla de probabilidad.

1.1 Definición del problema de problema de ruteo de vehículos capacitado

El problema puede ser definido formalmente como un grafo no dirigido $G = (V, E)$ donde $V = \{v_o, v_1, \dots, v_n\}$ es el set de vértices y $E = \{(v_i, v_j) | v_i, v_j \in V, i \neq j\}$ es un set de arcos. El depósito está representado por el véritce v_0 el cual posee m vehículos de reparto independientes con capacidad de carga igual Q para cumplir con la demanda q_i d,e n ciudades o clientes $i = 1,2, \dots n$ representadas por el set de n vértices $\{v_1, \dots v_n\}$. Hay un costo o distancia no negativa representada como matriz $C = (c_{ij})$ entre los clientes v_i y v_j definido en E. Una solución del CVRP sería la partición de $R_1, R_2, \dots, R_m$ en V representando las rutas de los vehículos donde cada ruta $R_i = \{v_{i0}, v_{i1}, \dots, v_{ik+1}\}$ donde $v_{ij} \in V$ y $v_{i0} = v_{ik+1} = 0$ (0 es el depot) satisfaciendo $\sum_{v_{ij} \in R_k} q_j \leq Q$. El costo del problema esta definido por la distancia recorrida de los vehículos: $\sum_{j=0}^{k} c_{jj+1}$

1.2 Modelamiento Matemático

Se expone a continuación la formulación de Wang y Lu [1].

1.2.1 Parámetros

$$d_{ij}: Distancia\ desde\ el\ nodo\ i\ al\ j$$
$$q_i: demanda\ del\ nodo\ i$$
$$Q: Capacidad\ máxima\ de\ los\ vehículos$$

1.2.2 Variables de decisión

$$x_{ij} = \begin{cases} 1 & Si\ se\ va\ del\ nodo\ i\ al\ j \\ 0 & dlc \end{cases}$$

$$y_{ik} = \begin{cases} 1 & Si\ la\ demanda\ del\ nodo\ i\ es\ atendida\ por\ el\ camion\ k \\ 0 & dlc \end{cases}$$

1.2.3 Función Objetivo

Se busca minimizar el costo total equivalente a la distancia recorrida por los vehículos en servicio.

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij}$$

1.2.4 Restricciones

(0) Naturaleza de variables

$$x_{ij} \in \{0,1\}, i = 1, \dots, n; j = 1, \dots n$$
$$y_{ik} \in \{0,1\}, i = 1, \dots, n; k = 1, \dots, m$$

(1) No se debe violar la capacidad de los vehículos

$$\sum_{i=1}^{n} q_i y_{ik} \leq Q, k = 1, \dots, m$$

(2) Restricción de Subtoures

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1, \quad S \subseteq \{2, \dots, n\}$$

(3) Un nodo cliente solo puede ser atendido por un vehículo. Del depot salen todos los vehículos.

$$\sum_{k=1}^{m} y_{ik} = \begin{cases} m & j = 0 \\ 1 & j = 1, \dots, n \end{cases}$$

(4) Restricción de balance. Si un vehículo sale de un nodo es porque entro y viceversa.

$$\sum_{i=1}^{n} x_{ij} = \sum_{i=1}^{n} x_{ji} = \begin{cases} m & j = 1 \\ 1 & j = 2, \dots, n \end{cases}$$

2 Revisión de Literatura

Los algoritmos genéticos fueron propuestos por primera vez por John Holland [4]. Estos hacen parte de una clase especial de métodos inspirados en la naturaleza. La idea por detrás de este algoritmo es modelar la evolución natural propuesta por Charles Darwin. Entre más adaptado este un individuo mayores probabilidades tendrá de sobrevivir y viceversa. Los GA son métodos de búsqueda informada capaces de explorar el espacio de soluciones por múltiples frentes a través de poblaciones. Una población de individuos representa soluciones tentativas que son mantenidas por generaciones. Cada individuo es una solución en particular y estos están codificados a través de cromosomas. Nuevos individuos son producidos a través de la combinación de miembros de la población a través de crossover y operadores de mutación y reemplazan a los individuos actuales a través de unas reglas [5].

Los métodos de crossover son responsables de producir los hijos al recombinar parte de la solución de los individuos padre para explorar el espacio de soluciones de forma efectiva y mejorar las soluciones. La mayoría de los métodos de crossover crean nuevos hijos usando un método estocástico que no tiene en cuenta la función objetivo. Otros si la tienen en cuenta y producen dos hijos: uno que mejora la. Función objetivo y otro que mantiene la diversidad del espacio de soluciones [6]. La mutación es requerida para explorar nuevos estados y ayuda al algoritmo a evitar óptimos locales [7].

Los algoritmos genéticos son combinados frecuentemente con otro tipo de heurísticas brindando muy buenos resultados. Hay numerosas aplicaciones en la literatura que generalmente se enfocan en usar un algoritmo en la explotación local de cromosomas, en la inicialización de una población bien estructurada o en el proceso de fabricación de individuos [1]. Prins [8] desarrollo un GA híbrido para resolver el VRP y hallo que heurísticas convencionales pueden crear una solución inicial bien estructurada.

Para encontrar una solución inicial varios autores han sugerido métodos que tienen en cuenta el clustering de nodos por sus propiedades axiales y de radio en comparación con el depot [9]. Debido a que se tienen que generasr varias soluciones iniciales, se deben usar métodos diversos que varíen de alguna forma las soluciones iniciales de una forma estructurada para garantizar diversidad. La codificación más utilizada para problemas como el TSP que son vistos como permutación es la representación de caminos. En esta codificación, la solución es codificada como una permutación de un set de enteros [10]. Usar esta codificación permite tener acceso a múltiples operadores de crossover que ya se han diseñado para este problema

La estructura de los cromosomas iniciales en GA y los parámetros usados involucrados en las probabilidades de crossover y mutación afectan de forma significativa el poder de solución de los GA. Sin embargo, estos parámetros dependen de los atributos del VRP, incluyendo el número de nodos y la distribución de ellos y el depot. Sin embargo, no hay a la fecha un método sistemático para deducir de forma precisa estos parámetros y más estudios son necesarios [1]. Mientras tanto se deben realizar simulaciones del tipo de poblaciones conseguidas a través de la variación de parámetros.

Entre los crossovers más conocidos se encuentran el Partially mapped crosover el cual escoge dos puntos P_1 y P_2 $(P_1 \leq P_2)$ de manera aleatoria uniforme. Estos puntos definen la sección a cortar e intercambiar entre dos cromosomas a través de intercambios posición a posición. Esto generalmente produce rutas infactibles al haber clientes repetidos y clientes sin atender. Es por esto por lo que, si se encuentra un número duplicado, este es intercambiado con el número ubicado en la misma posición del cromosoma recortado del otro individuo. Ahora, si el número reemplazado también se encontraba duplicado en el otro cromosoma, se debe hacer un mapeado de los números. Otro crossover es el de orden. En este se seleccionan dos puntos de crossover entre dos padres. Los hijos heredan los elementos entre los puntos

de crossover del primer padre. Los elementos restantes son heredados del segundo. El crossover uniforme usa tres pasos. Un vector binario del tamaño de los cromosomas es generado y con base en este se selecciona el elemento de la posición del padre 1 o 2 dependiendo del vector binario. Como métodos de mutación se encuentran generalmente Swap, reversa y Swap inverso. La mutación de Swap (1-OPT) selecciona dos genes aleatoriamente e intercambia sus posiciones. La mutación reversa escoge aleatoriamente dos puntos de corte e invierte el sentido de la ruta entre estas posiciones. [11]

Los métodos de crossover anteriormente mencionados son usados para todo tipo de problemas combinatorios donde se tenga una cadena de números o letras que representen la solución. Algunos autores han sugeridos crossovers específicos para el VRP que tengan en cuenta las propiedades de las rutas. Mulloorakam y Nidhiry [12] propone en crossover de Mejor costo de ruta (BCRC) y subruta Mapeada (SMCM). En el BCRC una ruta es seleccionada de cada padre aleatoriamente. Luego los clientes de las rutas seleccionadas son eliminados del otro padre. Posteriormente, estos clientes son reinsertados en las mejores posiciones posibles, uno por uno. En el SMCM, los últimos clientes de las rutas de los dos padres se convierten en la primera ruta de los hijos, sujeto a las restricciones de capacidad. Si se viola la restricción, se crea una nueva ruta. Ahora se remueven los clientes en las rutas formadas de alguno de los hijos del padre del otro hijo, para cada ruta. Luego, las rutas restantes de los padres son copiadas y los últimos nodos de las rutas remanentes son intercambiados e insertados satisfaciendo las restricciones. Este proceso se repute hasta que todas las rutas y sus clientes hallan sido intercambiados.

3 Metodología de solución

Para esta tarea implementaremos en Python un algoritmo genético para resolver el CVRP. En este algoritmo usaremos dos tipos de crossover para compararlos: el BCRC y el Partially Mapped Crossover. El método de BCRC fue implementado mientras que el PMC es obtenido del código de Damian Pirchio [13], el PMC también fue implementado pero no se tenía en cuenta la transitividad de los nodos repetidos y copiados, se adjunta esta implementación por aparte. Como método de mutación usaremos el Swap por su sencillez. Como método de inicialización crearemos rutas de forma aleatoria, otros posibles métodos podrían ser usar reglas geométricas basadas en el radio o inicializaciones sencillas como el vecino más cercano. Creemos que al formar rutas de forma aleatoria tendremos una mejor vista del espacio de soluciones a costa de un tiempo computacional más extendido para llegar a buenas soluciones. Por último, se usará el método de la ruleta para determinar los mejores cromosomas a ser reproducidos y la mutación y crossover estarán sujetos a probabilidades.

El fitness se midio con la formula:

$$fitness = \frac{100}{FO}$$

3.1 Pseudocódigo

Input: $G(V, E)$: $Grafo\ del\ problema$, d_i: $Demanda\ del\ nodo\ i$, Q: $Capacidad\ de\ cada\ vehículo$
p_c: $Probabilidad\ de\ crossover$, p_m: $Probabilidad\ de\ mutación$, MAX_{ITER}: $Número\ de\ generaciones$,
$BCRC$: $Método\ de\ crossover\ a\ utilizar$, $tamaño$: $Tamaño\ población$
Output: Mejor Solución

Algoritmo Genético:

$poblacion \leftarrow Inicializar\ población\ aleatoriamente(tamaño)$
Calcular Fitness de $poblacion$
For generacion =1 to MAX_{ITER}:
    For individuo = 1 to $tamaño$:
        $padre_A, padre_B \leftarrow Seleccionar\ padres\ de\ problación\ basados\ en\ su\ fitness$
    If aleatorio() $\leq p_c$:
        If BCRC:
            $ruta_A, ruta_B \leftarrow$ Seleccionar una ruta aleatoriamente de cada padre

Quitar nodos de las rutas seleccionadas del otro padre
Reintroducir clientes quitados en la mejor posición uno por uno
$hijo_A, hijo_B \leftarrow Calcular\ fitness\ y\ devolver\ Hijos$
Else:
$p_1, p_2 \leftarrow Generar\ dos\ posiciones\ aleatoria\ del\ cromosoma$
Intercambiar los substrings entre los padres
Mapear los clientes repetidos y hacer lista de intercambio
$hijo_a, hijo_b \leftarrow$ Arreglar los hijos
If aleatorio() $\leq p_m$:
$hijo_{swap} = Escoger\ hijo\ aleatoriamente$
Escoger dos clientes e intercambiar sus posiciones
Calcular Fitness y agregar a siguiente población
Retornar mejor solución de población

3.2 Instancias a utilizar

Se utilizaron las instancias más conocidas para este problema: la librería CVRPLIB [14]la cual contiene varios sets de instancias que difieren en su método de generación para ubicar los clientes: basado en clusters, uniforme, distribución geométrica o casos del mundo real. Se usaron las instancias P propuestas por Augerate en 1995 [15] las cuales fueron generadas de forma aleatoria y brindan varios tamaños que se asemejan a los encontrados en la vida real: desde 16 hasta 101 nodos.

Table 1 Descripción de las instancias utilizadas

Instancia	*Númrero de Clientes*	*Número de vehículos mínimos*	*Capacidad Máxima*	*óptimo (entero)*
P-n16-k8	15	8	35	450
P-n19-k2	18	2	160	212
P-n20-k2	19	2	160	216
P-n21-k2	20	2	160	211
P-n22-k2	21	2	160	216
P-n22-k8	21	8	3000	603
P-n23-k8	22	8	40	529
P-n40-k5	39	5	140	458
P-n45-k5	44	5	150	510
P-n50-k7	49	7	150	554
P-n50-k8	49	8	120	631
P-n50-k10	49	10	100	696
P-n51-k10	50	10	80	741
P-n55-k7	54	7	170	568
P-n55-k10	54	10	115	694
P-n55-k15	54	15	70	989
P-n60-k10	59	10	120	744
P-n60-k15	59	15	80	968
P-n65-k10	64	10	130	792
P-n70-k10	69	10	135	827
P-n76-k4	75	4	350	593
P-n76-k5	75	5	280	627
P-n101-k4	100	4	400	681

3.3 Análisis Estadístico

Para calibrar las soluciones se utilizaron los siguientes valores para los parámetros del algoritmo.

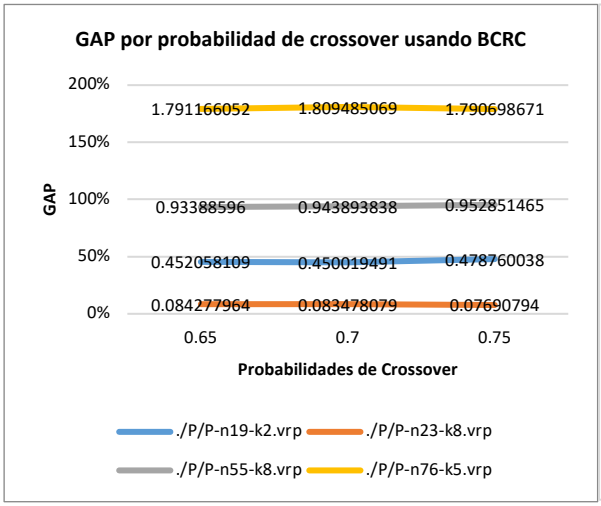
Parámetro	Valores a calibrar
Probabilidad de Crossover	0.65,0.7,0.75
Probabilidad de mutación	0.05, 0.1
Número de Generaciones	800,1200,1600,2000
Métodos de Crossover	BCRC & Partially Mapped Crossover
Tamaño de Población	70,100,130

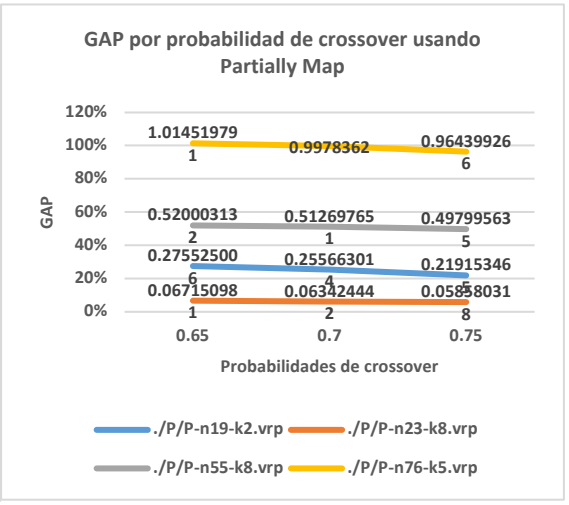
Se presentan a continuación la comparación entres los diferentes parámetros de manera individual. Para cada parámetro se mostrará el desempeño en GAP para cada valor probado del parámetro. Debido a que hay dos métodos de crossover se harán dos gráficas por separado. Se usaron cuatro instancias que

representan diferentes tamaños y capacidades máximas del set de instancias utilizadas.

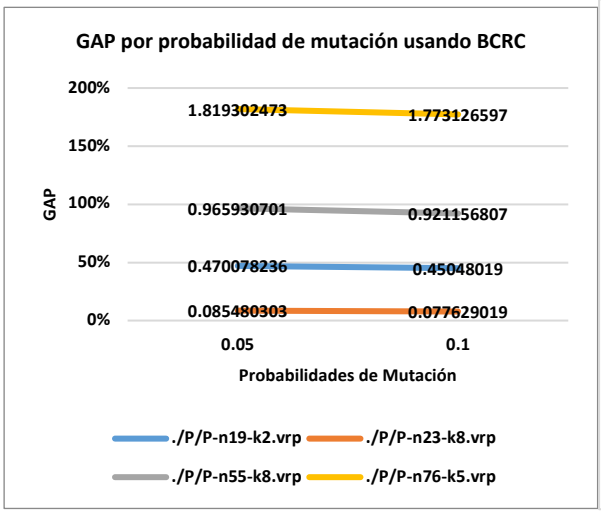
El resultado más claro del análisis de parámetros es la preferencia por el operador de crossover Partially Mapped el cual obtuvo en las instancias más grandes un GAP hasta dos veces menor. Es por esto por lo que se usará este método de crossover para resolver todas las instancias. Con respecto a los otros parámetros, observamos que la probabilidad de crossover que mejores resultados obtiene es 0.75 para la mayoría de las instancias, igualmente la probabilidad de mutación con mejores resultados es 0.1. Por otro lado, los resultados del análisis muestran que en general para instancias muy grandes, se requiere contar un tamaño de población grande y un alto número de generaciones para reducir el GAP, esto es más necesario en instancias grandes donde la diferencia es crucial. Decidimos así usar un número de generaciones igual a 1200 para instancias menores a 50 clientes y 2000 para instancias mayores o iguales a 50 clientes. Por otro lado, usaremos 130 individuos para todas las instancias.

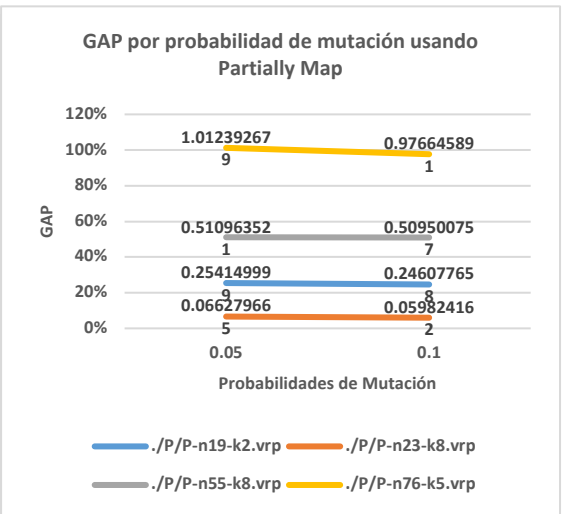
3.3.1 Análisis de la probabilidad de crossover



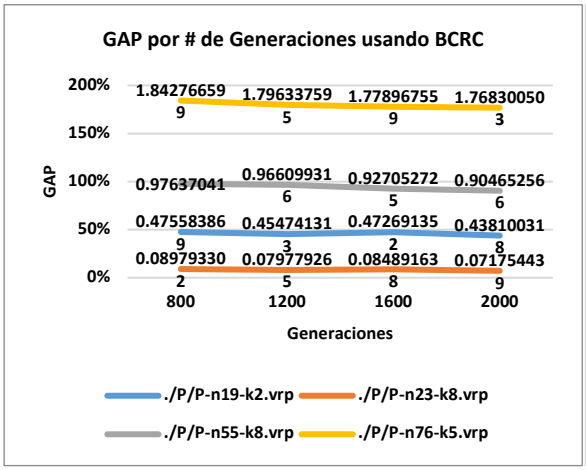


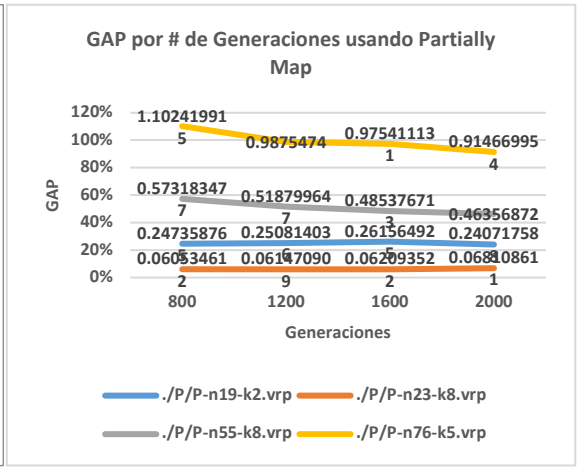
3.3.2 Análisis de la probabilidad de mutación



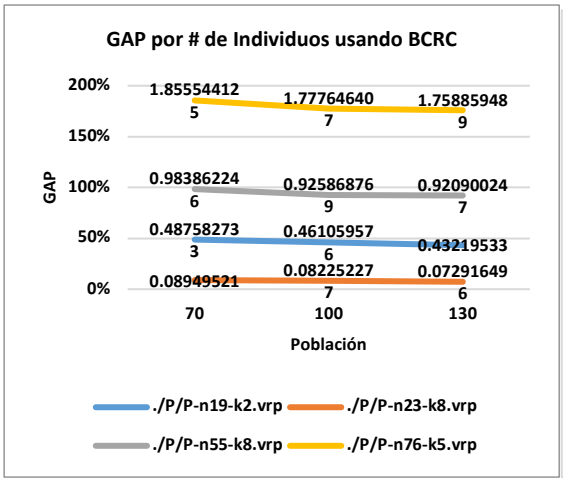


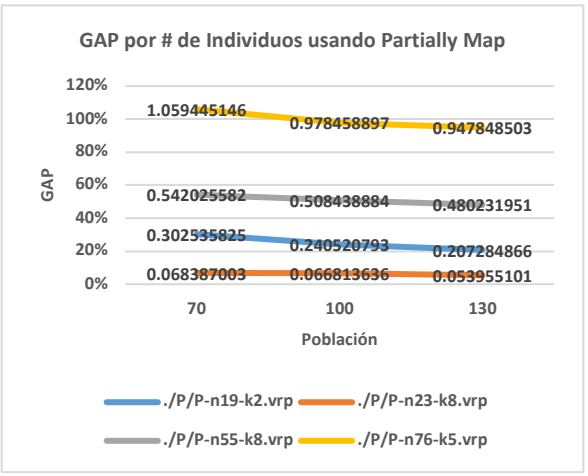
3.3.3 Análisis de el número de generaciones





3.3.4 Análisis del número de Individuos





3.4 Detalles de Implementación

Se implementaron los algoritmos en Python 3.8 usando una conceptualización de clases para priorizar el ahorro de memoria. La implementación recibe como parámetros el

// python HW4.py
// para correr heurística con $p_c = 0.75, p_m = 0.1, MAX_{ITER} = 800; BCRC, poblacion = 100$
$ python HW4.py 0.75 0.1 800 1 100

También se genero un archivo ejecutable para equipos que no tengan Python instalado para equipos tipo Linux. Las instancias se pueden correr con un archivo .sh para correr todos los experimentos acá presentados. Se incluyo un README en los soportes de esta tarea que indican como ejecutar estos archivos.

4 Experimentación y Análisis de Resultados

Estas fueron corridas en un procesador dual core “Broadwell” 2.7 GHz Intel “Core i5” (5275U) con 8 GB 1866 MHz LPDDR3 SDRAM corriendo macOS Catalina 10.15.6 con Python 3.8. Dada la aleatoriedad de los algoritmos, estos fueron ejecutados 30 veces por algoritmo para la experimentación y discusión de resultados.

Hay que tener en cuenta que los resultados a continuación presentados son hechos con el crossover Partially Map el cual le gano al BCRC.

4.1 Simulación

Para simular el desempeño del algoritmo genético y de las poblaciones producidas se gráfico el puntaje de fitness promedio de los individuos de la población de cada generación. Al revisar la mejora del fitness con el paso de las generaciones notamos que en las primeras 500 generaciones el mejoramiento fue exponencial. En las siguientes generaciones se disminuye el ritmo de mejora hasta llega a la generación 2300 donde la población queda estancada a 1.0 de GAP del óptimo.

Podemos explicar este estancamiento debido a la sencillez del algoritmo. Se requieren algoritmos híbridos o mecanismos intermedios que amplíen el espacio de solución y no permitan que el algoritmo se quede en un óptimo local. El comportamiento de las poblaciones es el esperado y, con respecto a la solución inicial se mejora a más del doble del fitness en la mejor solución encontrada. El siguiente gráfico muestra la media $\pm$ la desviación estándar.

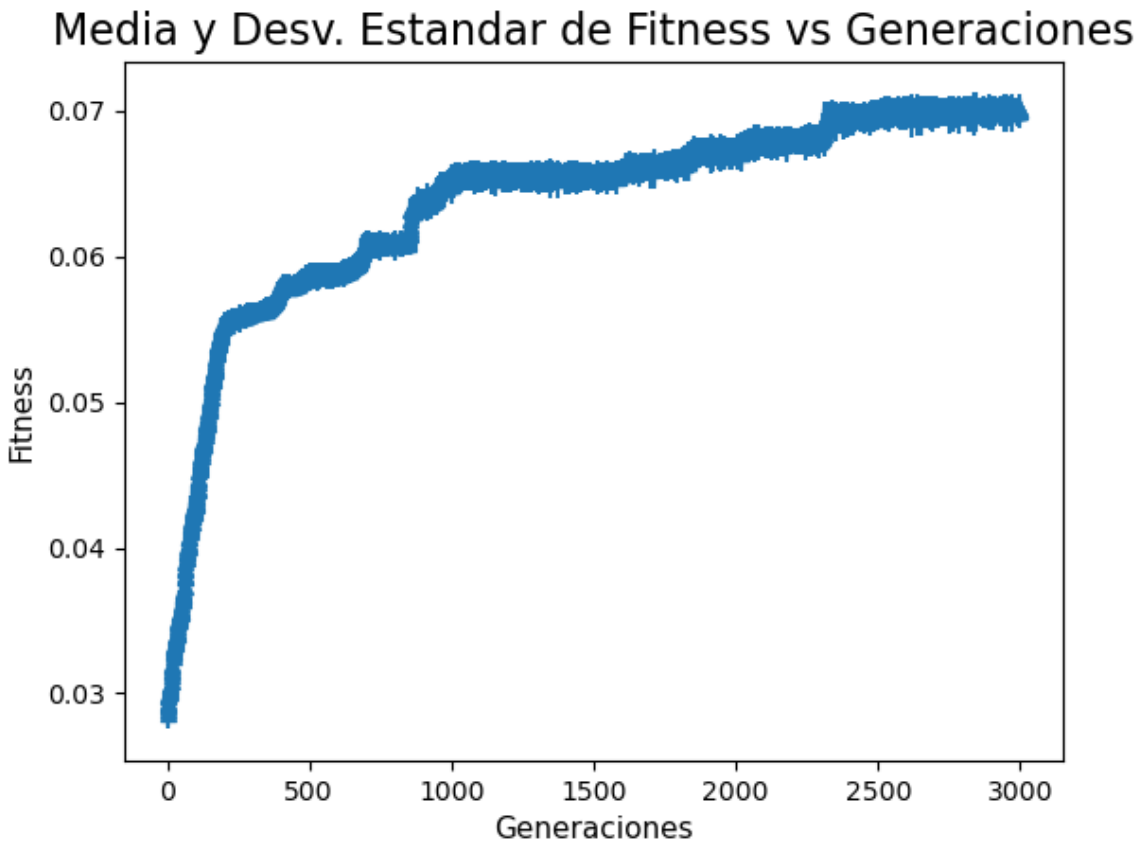


Figure 1 Media y Desviación Estándar del fitness de las generaciones

4.2 Resultados de las instancias

Las instancias con nodos menores a 500 obtuvieron un GAP menor al 20% lo cual es consistente con lo esperado por una Meta heurística. Para instancias mayores a 50 nodos, los GAPS se incrementan hasta llegar en promedio al 40%. Para las instancias más difíciles, este GAP es del 150 % lo cual refleja la dificultad del problema, pero también la necesidad de expandir las capacidades del algoritmo con otro tipo de heurísticas en forma de un híbrido. En tiempos computacionales, el algoritmo toma hasta un minuto en las instancias más grandes para resolver con los parámetros suministrados. Esto se debe en

parte a la necesidad de revisar el cumplimiento de las restricciones en cada momento del algoritmo y su consecuente costo en tiempo.

Al analizar la simulación de las generaciones vemos que las soluciones se quedan estancadas en un óptimo local, por lo que es necesario usar un mecanismo adicional para generar diversidad como un método de Simualted Annealing o búsqueda Tabu como es generalmente usado en la literatura. También es cierto que el uso de una inicialización inteligente basada en la situación del problema podría mejorar los tiempos computacionales del algoritmo al generar buenas soluciones en un tiempo más corto.

A continuación, mostramos los resultados detallados para todas las instancias P de Augerat.

Table 2 Resultados detallados de las instancias

MÍNIMO DE VEHÍCULOS	NÚMERO DE NODOS	GAP PROMEDIO	SOLUCION PROMEDIO	TIEMPO PROMEDIO (S)
8	16	0.79%	454.094	10.375
8	22	3.42%	624.192	14.195
8	23	4.93%	555.759	14.247
8	50	29.42%	817.200	36.688
8	55	41.54%	832.681	37.272
15	55	12.37%	1112.021	54.410
15	60	23.11%	1192.180	45.348
2	19	16.27%	246.871	10.507
2	20	21.44%	262.860	12.772
2	21	17.00%	247.445	17.168
2	22	20.88%	261.486	12.785
10	50	26.39%	880.242	44.430
10	51	31.34%	973.727	40.727
10	55	32.62%	920.990	45.661
10	60	37.65%	1024.702	44.220
10	65	42.41%	1128.541	48.551
10	70	48.78%	1230.791	50.717
5	40	41.08%	646.530	17.476
5	45	45.63%	743.207	19.940
5	76	79.29%	1124.789	49.127
7	50	43.57%	795.968	42.595
7	55	45.49%	826.903	47.305
4	76	89.73%	1125.589	49.858
4	101	138.45%	1624.427	69.835

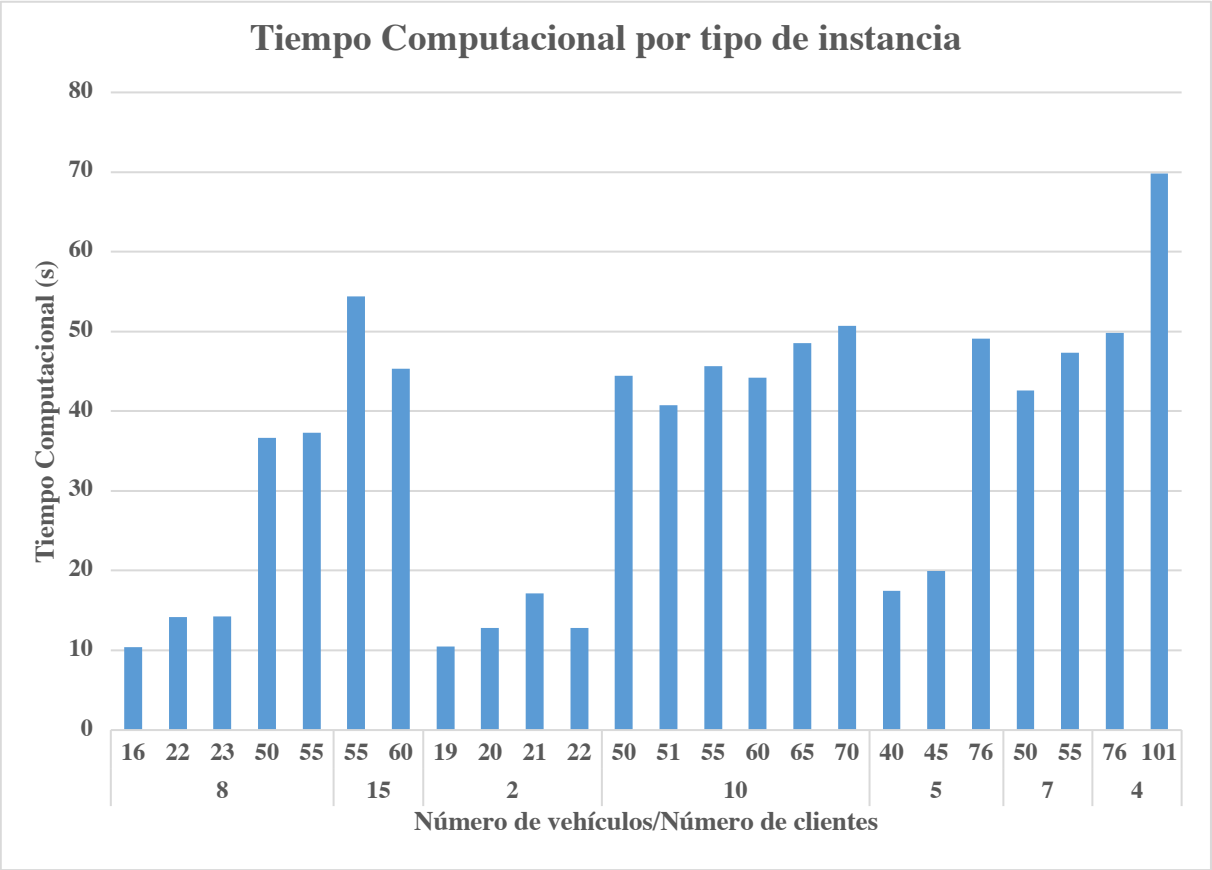


Figure 2 Tiempo Computacional por tipo de instancia

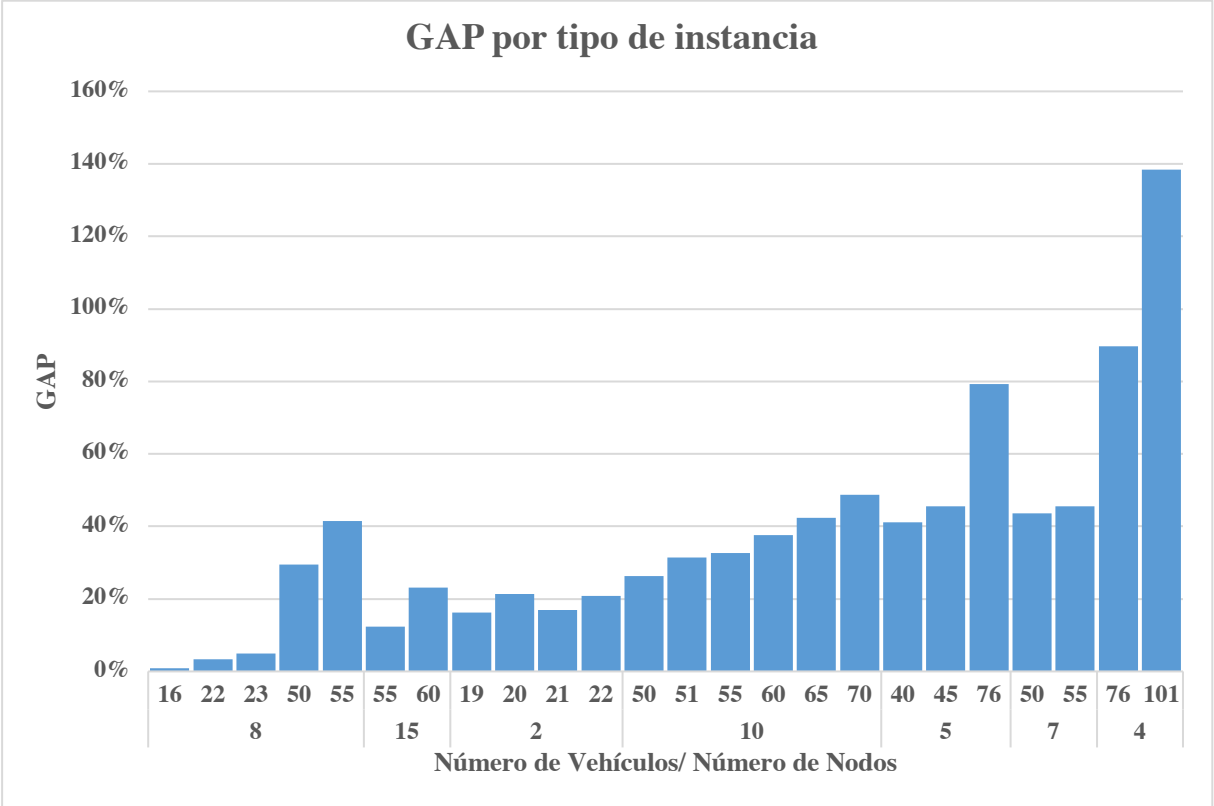


Figure 3 GAP por Tipo de instancia

5 Conclusiones

- El algoritmo genético es una metaheurística capaz de resolver cualquier tipo de problema y logra resolver el CVRP de forma efectiva para instancias pequeñas. Para instancias pequeñas se encontraron soluciones con GAP menores al 20% siendo competitivo para estas instancias. Para instancias realistas de más de 50 nodos el desempeño del algoritmo disminuye y encuentra soluciones de GAP del 40% en promedio.
- El método de crossover BCRC, a pesar de ser diseñado para el VRP, no logra mejorar los resultados del método Partially Mapped Crossover el cual logra GAPS 2 veces menores al BCRC. Esto se debe posiblemente a que el PMC es un método esencialmente genético mientras que el BCRC puede considerarse un método de mutación sofisticado que involucra y cruza dos padres.
- El algoritmo realizado logra aumentar el fitness de forma constante en cada generación y se queda en un óptimo local después de más de 500 generaciones. Es necesario en el futuro complementar este algoritmo con otras técnicas que produzcan diversidades o varias poblaciones con distintos métodos de inicialización para evitar caer en óptimos locales.
- Se encontró, a través de varias simulaciones que los mejores parámetros de probabilidad de crossover y mutacion son 0.75 y 0.1 respectivamente. El número de generaciones e individuos dependen del tamaño del problema y para instancias grandes se usaron 2000 generaciones y 130 individuos.

6 Referencias

[1] C.-H. Wang y J.-Z. Lu, «A hybrid genetic algorithm that optimizes capacitated vehicle routing problems,» *Expert Systems with Applications,* vol. 36, pp. 2921-2936, 2009.

[2] Lawrence, Bodin, Bruce y Golden, «Classification in vehicle routing and scheduling,» *Networks,* vol. 11, pp. 97-108, 1981.

[3] D. E. Goldberg, enetic algorithms in search, optimization and machine learning, MA: Addison-Wesley, 1989.

[4] J. Holland, Adaptations in Natural and Artificial Systems, Ann Arbor: The University of Michigan Press, 1975.

[5] D. Whitley, «Next Generation Genetic Algorithms: A User's Guide and Tutorial,» de *Handbook of Metaheuristics*, Fort Collins, Springer International Publishing, 2019, pp. 245-290.

[6] H. Nazif y L. S. Lee, «Optimised crossover genetic algorithm for capacitated vehicle routing problem,» *Applied Mathematical Modelling,* vol. 36, pp. 2110-2117, 2012.

[7] B. Koohestani, « A crossover operator for improving the efficiency of permutation-based genetic algorithms,» *Expert Systems With Applications,* vol. 151, nº 113381, 2020.

[8] C. Prins, «A simple and effective evolutionary algorithm for the vehicle routing problem.,» *Computer and Operations Research,,* vol. 31, pp. 1985-2002, 2004.

[9] J. J. Bentley, «Fast algorithms for geometric traveling salesman problem,» *ORSA Journal on Computing,* vol. 4, pp. 387-411, 1992.

[10] M. S. K. Du, Search and optimization by metaheuristics: techniques and algorithms inspired by nature, Springer International Publishing , 2016.

[11] M. Haj-Rachid, W. Ramdane-Cherif, P. Chatonnay y C. Bloch, «Comparing the performance of genetic operators for the vehicle routing problem,» de *Management and Control of Production Logistics*, Coimbra, Portugal, 2010.

[12] A. T. Mulloorakama y N. M. Nidhiryb, «Combined Objective Optimization for Vehicle Routing Using Genetic Algorithm,» *Materials Today: Proceedings,* vol. 11, pp. 891-902, 2019.

[13] D. Pirchio, «Flowshop-GA,» https://github.com/damianpirchio/Flowshop-

GA/blob/master/metaheuristicas.py, 2014.

[14] L. Xavier, *CVRPLIB,* http://vrp.atd-lab.inf.puc-rio.br/index.php/en/, 2020.

[15] P. &. B. J. M. &. B. E. &. C. A. &. N. D. &. R. G. Augerat, «Computational results with a branch and cut code for the capacitated vehicle routing problem,» ISTITUTO DI ANALISI DEI SISTEMI ED INFORMATICA, Roma, 1998.

[16] S. Faiz, S. Krichen y W. Inoubli, «A DSS based on GIS and Tabu search for solving the CVRP: The Tunisian case,» *The Egyptian Journal of Remote Sensing and Space Science,* vol. 17, nº 1, pp. 105-110, 2014.

[17] T. Öncan, I. K. Altınel y G. Laporte, «A comparative analysis of several asymmetric traveling salesman problem formulations,» *Computers & Operations Research,* vol. 36, pp. 637-654, 2009.