OPTIMIZACIÓN BASADA EN LA MIGRACIÓN DE LAS AVES

Aplicada al problema de ruteo vehicular

23 DE MAYO DE 2017

Estudiante: Barbero Alisandroni Juan José
Tutora: Dra. Gabriela F. Minetti

Agradecimientos:

Principalmente a mi familia. Mi madre Marcela y mi padre Ricardo, por hacer que el estudio sea el objetivo principal, por enseñarme a enfrentar obstáculos y mantenerme en mi camino en momentos de flaqueza. A mi abuelo Rubén, por enseñarme el valor de la palabra y el compromiso. A mi abuela Margarita, por mantener la vela que me guió durante el estudio y dedicarme un lugar en su mesa durante varios años. A mis hermanas Anaclara y Valentina, por los momentos de distracción y el cariño compartido. A todos, gracias por el apoyo brindado y el que me brindarán. También quiero darle las gracias a mi tutora Gabriela por su apoyo, predisposición y enseñanzas, y especialmente por su paciencia durante el desarrollo de este trabajo. Finalmente es mi deseo agradecer a todos los que de alguna forma influyeron en que yo esté en este momento y lugar de mi vida.

Índice

Índice de tablas

Índice de figuras

Capítulo 1: Introducción

Optimizar significa encontrar la mejor solución que equilibre la ecuación costo-beneficio en favor del beneficio obtenido, en ese punto podemos decir que realmente se halló una solución óptima. Esto es una tarea simple cuando los problemas presentan un número reducido de variables y pueden ser resueltos en tiempo polinomial. Sin embargo en la mayoría de los problemas existentes, la cantidad de variables que intervienen hace muy difícil o imposible encontrar una solución óptima en un tiempo razonable. Este tipo de problemas se denominan NP-duros *(NP-hard)*. A causa de la cantidad de variables involucradas y el tiempo requerido para analizar todas las posibles soluciones resulta necesario implementar otras herramientas con el fin de agilizar la búsqueda. Tales herramientas son denominadas heurísticas. Éstas permiten usar el conocimiento y experticia en un problema concreto para hallar rápidamente una solución. Pero este método tiene por desventaja que está fuertemente ligado al problema que se busca resolver. Para evitar tal dilema es necesario recurrir a otras herramientas, más allá de las heurísticas, que permitirán resolver una gama más amplia de problemas, éstas son las metaheurísticas. Existen diferentes tipos de metaheurísticas que han sido desarrolladas en los últimos años, principalmente se organizan en dos grandes grupos. Basadas en trayectoria (cuyo trabajo consiste en manipular una única solución hasta obtener una estructura óptima o con un error aceptable, en caso de no existir un óptimo). Y basadas en población (las cuales evolucionan un conjunto de soluciones durante una cierta cantidad de iteraciones hasta encontrar el mejor individuo que resuelva el problema en cuestión).

Las metaheurísticas han demostrado su potencial para la resolución de problemas con características similares, y es una herramienta que se ha mostrado útil para abordar la resolución del problema en consideración. Cuentan con una amplia bibliografía y aplicaciones documentadas donde se puede afirmar su versatilidad, ya que es posible combinarlas con heurísticas del problema a resolver y/o con otras metaheurísticas obteniendo en algunos casos mejoras considerables.

Por otro lado, en los últimos años han surgido nuevas técnicas metaheurísticas que resultaron prometedoras al momento de resolver otros problemas que cuentan con una arquitectura similar al que se planea resolver. Una de ellas es el algoritmo propuesto por Duman [1] basado en la migración de las aves *(Migrating Birds Optimization Algorithm* o *MBO)*. Este algoritmo, que recientemente vio la luz, ha demostrado su valía resolviendo problemas de transporte marítimo [2], secuenciación de tareas [3], por mencionar algunos, y con esto aporta la motivación necesaria para ponerlo a prueba con el problema de ruteo vehicular con capacidad.

Esta variante del problema del viajante de comercio es un caso de amplio estudio debido a su gran incidencia en el mundo empresarial ya que aumentar el margen de ganancias y la productividad es el objetivo primordial de todo negocio. En esto radica el interés en su estudio y la razón por la cual en los últimos años el problema se ha ramificado en distintas versiones. Algunas a mencionar puede ser, el problema de ruteo vehicular con ventanas de tiempo, con depósitos múltiples, dependiente por áreas, entre otros; cada una de ellas con su respectivo conjunto de restricciones.

En el trabajo propuesto se pretende implementar el algoritmo MBO y aplicarlo al problema mencionado. Paralelamente a esto, se pone en curso la tarea de recabar información necesaria sobre el estado del arte en lo relacionado al problema y al algoritmo. Y una vez concluida la implementación, diseñar y llevar a cabo el experimento propuesto con el fin de documentar los resultados obtenidos y realizar un análisis de desempeño.

El resto del documento se organiza de la siguiente manera, en el capítulo 2 se halla una breve reseña sobre el mecanismo de migración que emplean las aves, seguido de la presentación del algoritmo y la manera en que interactúan sus parámetros. En el capítulo 3 se realiza un estudio del estado del arte en cuanto al problema seleccionado, junto con la presentación formal del problema. El capítulo 4 instruye sobre la implementación del algoritmo para el problema de ruteo vehicular. En el capítulo 5 puede hallarse una reseña sobre el diseño del experimento, seguido de una sección que explica la configuración de los distintos parámetros y presenta los valores óptimos de los mismos. Sobre el final de éste se realiza un análisis de resultados desde dos puntos de vista diferentes, por un lado se evalúa la calidad del algoritmo mientras que por otro lado el foco se coloca en el desempeño de éste. Finalmente, en el capítulo 6 se presentan las conclusiones del trabajo, así como también los posibles trabajos futuros.

Capítulo 2: Optimización basada en la migración de las aves

En este capítulo se describe cómo las aves realizan la migración, esto permitirá luego entender el algoritmo de optimización basado en este proceso natural. Seguidamente se explica el algoritmo y el modo en que sus parámetros interactúan. Finalmente, se presenta un estudio de las investigaciones publicadas en la literatura sobre MBO.

2.1. Migración de las aves

Al momento de emprender la migración el objetivo principal de las aves es reducir el esfuerzo requerido para recorrer grandes distancias, por esto adoptan distintas formaciones de vuelo. Este trabajo en particular se enfoca en la estructura de vuelo en "V", cuyo nombre surge de la similitud que presenta el modo en que se posicionan las aves con la letra en cuestión. En esta formación el líder de la bandada se ubica en el vértice de la formación y las aves restantes se distribuyen de forma escalonada en dos ramas detrás de él.

Antes de comentar el mecanismo por el cual esta formación disminuye el esfuerzo requerido por el ave, es necesario realizar una breve mención de la mecánica de vuelo que poseen estos animales. Las alas de un pájaro actúan como una superficie sustentadora, esto es así debido a la forma en que el aire circula a través de ellas. El esquema del ala puede verse en la Figura 01. Si hacemos un corte de perfil en el ala podemos ver que el aire que fluye por encima debe recorrer una distancia mayor a diferencia del aire que lo hace por debajo, esto ocasiona que la presión ejercida en la parte superior sea menor en comparación con la presión en la parte inferior y posibilita la elevación del pájaro. La fuerza necesaria para generar este momento de elevación se conoce como poder inducido.

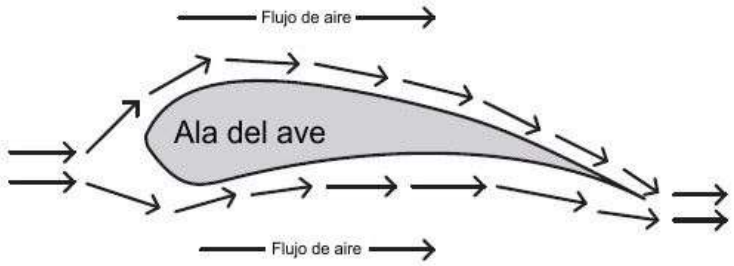

Figura 01: Corte transversal del ala de un ave.

Para un ave solitaria la clave para obtener una buena sustentación es ganar velocidad, como consecuencia habrá una disminución de la presión en la parte superior del ala y se producirá una mayor elevación. Pero esto requerirá de un mayor poder inducido e implica que el ave se cansará en un corto periodo de tiempo. Lo cual ocasionará que no pueda recorrer grandes distancias, o lo haga en un tiempo excesivamente largo ya que tendrá que realizar múltiples paradas. Es aquí donde, tanto los años de evolución como la naturaleza, actúan para

optimizar su desempeño en la tarea de recorrer extensos trayectos basándose en la sinergia de grupo.

Cuando un pájaro emprende el vuelo, la turbulencia que desprenden sus alas genera dos vórtices de aire que fluye en forma circular. De esta forma se generan dos sectores con corrientes ascendentes y una zona central con corrientes descendentes, como puede apreciarse en la Figura 02. Estas corrientes ascendentes actúan en beneficio del siguiente miembro de la bandada, brindándole una mayor sustentación y reduciendo el poder inducido necesario para mantener el vuelo [4].

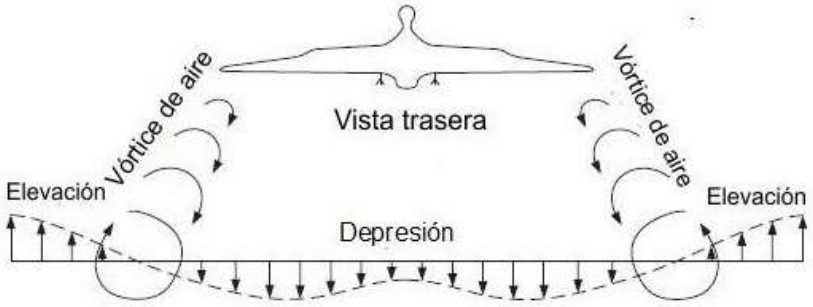

Figura 02: Esquema del vórtice de turbulencia creado por las alas del ave.

Existen varios estudios en los cuales se analizan los distintos parámetros intervinientes en la formación en “V”, éstos se esquematizan en la Figura 03. Uno de ellos es la distancia que mantienen las aves entre ellas (denotado como Solapamiento), en el estudio realizado por Lissaman y Schollenberger [5] se estableció que a medida que las aves se aproximan unas a otras y el volumen de la bandada aumenta también se incrementa el ahorro de energía o, lo que es igual, se reduce el esfuerzo.

Otro parámetro que puede influir en el desempeño de la bandada es la profundidad con la que se ubican los distintos miembros en relación con el líder, la profundidad óptima está establecida como dos veces el ancho del ala [6]. Concretamente la profundidad con la que estos animales se posicionan está determinada por el ancho del ala, la separación que mantengan entre sí y el ángulo de apertura, el cual provee un línea de visión confortable entre las aves.

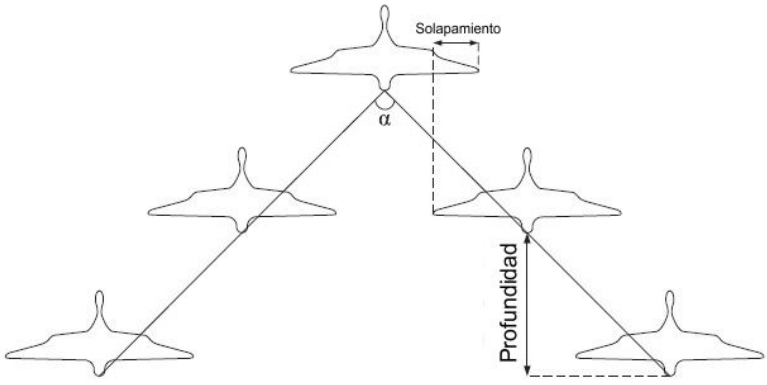

Figura 03: Esquema de la bandada junto con los parámetros dominantes.

Como se ha establecido, en la formación en “V”, el pájaro líder es el que se ubica en el frente y también es el que realiza el mayor esfuerzo. Es razonable pensar que a medida que bajamos en la jerarquía de la formación las aves ubicadas a una profundidad mayor se ven mayormente beneficiadas que aquellas que se ubican próximas al líder. Sin embargo hay estudios que establecen que el ahorro de energía que obtienen las aves, exceptuando al líder, es el mismo [7] o un tanto mayor para las aves ubicadas en el medio [8]. También es sabido que luego de cierto tiempo el líder habrá agotado sus fuerzas y será reemplazado por otro integrante de la bandada, mientras que éste se ubicará al final de la misma.

A pesar que en teoría, mientras mayor sea el número de aves en la bandada mayor será el ahorro de energía [5], en la práctica este número está limitado [4, 8]. Una explicación a esto es que a medida que el tamaño aumenta se hace más difícil para los pájaros en las posiciones más rezagadas mantener la línea y por consecuencia no se ven beneficiados por la formación.

2.2. Optimización basada en la migración de las aves (MBO)

Esta metaheurística toma como referencia el comportamiento de las aves durante sus largos viajes migratorios y lo reformula en un algoritmo de optimización. Cada solución desarrollada en la ejecución del algoritmo es la contraparte de un pájaro en la formación de vuelo en “V” y se ve beneficiada por la solución que tiene enfrente.

El proceso inicia con un número determinado de soluciones (aves) en donde la primera solución representará al pájaro líder de la bandada, a partir de él se generará todo un nuevo vecindario de soluciones y lo mismo se hará con el resto de las soluciones que componen la formación. Al momento de evaluar, la solución líder sólo podrá comparar su valor de fitness en contraste con su propio vecindario de soluciones. En caso de hallar una mejor opción, la existente será reemplazada y las mejores soluciones restantes serán cedidas a su vecino inmediato en la agrupación. Este individuo también generará un vecindario de soluciones pero en menor medida ya que cuenta con el beneficio de la solución anterior (note cómo esto corresponde con el mecanismo de beneficio obtenido por las aves que se trató en la sección anterior). A partir de la suma de soluciones cedidas con las generadas se obtiene un nuevo vecindario, en el cual se va a buscar el mejor candidato. En caso de que sea uno distinto al que ya está seleccionado se procede a intercambiarlo, y nuevamente el subgrupo de mejores soluciones se transfiere al siguiente individuo. Este procedimiento se mantiene por una cantidad determinada de *tours* (un tour en MBO es el tiempo durante el cual un ave lidera la bandada). Al alcanzar dicha cantidad, la solución líder pasa al final de la formación y es reemplazada por alguna de las dos soluciones que la siguen. El algoritmo se detiene luego de cumplir con un número estipulado de ciclos (ver Algoritmo 01).

```
1.  Generar aleat. n sol. iniciales      //n: número de soluciones iniciales.
2.  Colocarlas en una formación en V hipotética de forma arbitraria
3.  i = 0
4.  mientras (i < K)                     //K: N° máximo de soluciones
5.      para (j = 0; j < m; j++)         //m: número de iteraciones durante los cuales la solución líder ocupará ese rol
6.          Tratar de mejorar la solución líder generando k nuevas soluciones a partir de ella y evaluándolas
7.          i = i+k                      //k: cantidad de soluciones que formarán parte del vecindario
8.          para cada solución s_r perteneciente a la bandada (excepto el líder)
9.              tratar de mejorar s_r evaluando (k-x) vecinos generados a partir de ella y x vecinos cedidos por la solución anterior.
10.             i = i + (k –x)           //x: cant. de sol. pertenecientes a un vecindario compartidas con la prox. solución
11.         fin para
12.     fin para
13.     mover al líder al final de la formación y colocar una de las soluciones siguientes en la posición principal
14. fin mientras
15. devolver la mejor solución de la bandada
```

Algoritmo 01: Optimización basada en la migración de las aves.

Este algoritmo trata las soluciones como una bandada migratoria que emprenden el vuelo en una formación en "V", por lo cual el primer paso consiste en generar de forma aleatoria las *n* aves que compondrán la formación y ubicarlas de manera hipotética en forma de "V". A partir de estas soluciones iniciales, el algoritmo comenzará la exploración del espacio de búsqueda, al generar para cada solución, *k* nuevas soluciones que formarán el vecindario. Se comienza por la solución líder, se crea su vecindario y se evalúa cada solución en dicho vecindario. En caso de que alguna de ellas se ajuste mejor que aquella que se encuentra seleccionada, se procederá a cambiarla por el elemento de mejor ajuste. Seguidamente, se seleccionan los *x* mejores candidatos del vecindario y se transfieren a la siguiente solución de la formación. En este punto es donde entra en juego el mecanismo de beneficio ya citado. Dicho mecanismo actúa posibilitando que la solución siguiente tenga que generar una menor cantidad de soluciones para cubrir el tamaño del vecindario, al mismo tiempo que se le transfieren buenos individuos que permiten continuar la búsqueda por áreas más prometedoras del espacio de búsqueda. Al igual que sucedió en el caso de la solución líder, con los seguidores se procede a evaluar las $k - x$ soluciones generadas y realizar los cambios según el ajuste de cada una si fuera necesario. Esto ocurre durante una cantidad *m* de iteraciones, que tienen su analogía en el tiempo en que el líder podrá mantener esa posición antes de caer físicamente agotado y tener que trasladarse hacia alguna de las dos ramas que lo secundan. Luego de que el líder sea relevado por otra solución el proceso inicia nuevamente y se repetirá en su completitud hasta que se alcance la condición de finalización. La cual será alcanzada al generar un número *K* de soluciones, en ese momento el algoritmo retornará la mejor solución hallada.

2.3. Interacción de los parámetros de MBO

De lo expuesto anteriormente se puede deducir cómo el algoritmo se estructura en cuatro etapas claves, *inicialización*, *optimización del líder*, *optimización de los seguidores y reemplazo del líder*. En cada una de estas etapas troncales, los parámetros de inicialización tienen una incidencia clave en la forma en que se desempeñará el algoritmo.

En la etapa de inicialización primero se configuran los parámetros necesarios tales como, la cantidad de soluciones iniciales *(n)*, el tamaño de los vecindarios de soluciones *(k)*, la

cantidad de soluciones compartidas entre individuos consecutivos en la formación *(x)*, el número límite de iteraciones durante las cuales el líder encabezará la formación *(m)* y la cantidad de puntos del espacio de búsqueda evaluados *(K)*. A continuación se procede a generar las soluciones que serán el punto de partida de la búsqueda; en este punto puede hacerse una presunción poco exacta del tamaño de *n*. Es lógico que la cantidad de soluciones iniciales no puede ser pequeña, porque la solución que inició como líder de la formación volverá a repetir su posición en un periodo corto de tiempo. Es decir que, para un número dado de iteraciones, se tendrá una frecuencia de rotación del líder mayor cuando se trabaja con bandadas pequeñas, esto resultará en reiterar la búsqueda por sectores del espacio en los cuales ya se realizó. En contraste, si se trabajase con bandadas más numerosas la frecuencia en que se releva el líder será menor. De este hecho puede inferirse que, al trabajar con bandadas pequeñas contamos con una cantidad limitada de puntos de partida y por lo tanto se acotará la búsqueda pudiendo existir sectores del espacio que nunca serán evaluados. Si trasladamos lo expuesto a la naturaleza podemos ver como la distancia recorrida por bandadas reducidas es mucho menor que la distancia transitada por formaciones de mayor tamaño. Entonces, es lógico pensar que un número grande de soluciones iniciales es recomendable, pero en las líneas siguientes vamos a exponer por qué esto tampoco es aconsejable. Al existir un gran número de elementos de partida que conducen la exploración, la posibilidad de encontrar buenas soluciones crece a costa de un mayor esfuerzo computacional en todas las etapas del algoritmo. Lo cual se traduce en un mayor número de evaluaciones, de seguidores que aplican la búsqueda local, y en general, de soluciones que participan en el proceso migratorio. Si a esto se suma el hecho de tener una gran cantidad de focos de búsqueda es muy probable que los sectores de búsqueda individuales se solapen exageradamente, perjudicando el desempeño. Nuevamente si nos remitimos a la naturaleza las grandes bandadas cuando superan la cantidad óptima de integrantes se dividen y forman dos bandadas independientes, demostrando como no siempre la fuerza bruta es la solución y valores más modestos producen mejores resultados.

Una vez culminada la inicialización se procede a la optimización del individuo líder. Para lo cual se genera un vecindario de soluciones de tamaño *k* y se evalúa cada una de ellas para determinar si poseen un mejor ajuste en comparación con el elemento que les dio origen. El punto clave de esta etapa es el valor de *k* y, puede verse que, el efecto producido por este parámetro en el desempeño del algoritmo forma una curva acampanada, tal como ocurrió con *n* en la etapa anterior, seguidamente se analizara la causa de esto. Puede decirse que *k* es inversamente proporcional a la velocidad de vuelo de la bandada. Lo cual significa que si se le asigna valores bajos a *k* esta bandada hipotética se moverá a una mayor velocidad por el espacio de búsqueda relegando sectores del espacio sin inspeccionar. Tal como ocurre con las bandadas reales a mayor velocidad es más difícil apreciar el paisaje. Es posible suponer que esto afectará la velocidad de convergencia del algoritmo haciendo que encuentre sectores prometedores dentro del espacio de búsqueda mucho más rápido, pero se atascará en óptimos locales. Como contraparte, si a *k* se le asignan valores grandes implicará que la bandada realizará su viaje por el espacio de soluciones de una forma muy lenta, y por eso el tiempo de convergencia aumentará. En otras palabras, el volumen de soluciones que se evaluará será mayor y el tiempo requerido para encontrar una región prometedora también se incrementará.

Ya contando con los vecindarios de soluciones del líder creados y evaluados, y colocado el mejor candidato de todo el conjunto (líder más vecindario) en la posición de líder, es hora

de hacer uso del mecanismo de beneficio característico de esta metaheurística. El parámetro a considerar ahora será el valor de *x* que representa la cantidad de soluciones que se van a transferir al siguiente miembro de la bandada. Esta cifra tiene su contraparte natural en el grado de solapamiento que adoptan las aves al formar la bandada y que, según se demostrará en la sección 5.2, debe tomar magnitudes bajas. Teniendo todo esto en cuenta es posible hacer un análisis previo de cómo afectará al desempeño del algoritmo los cambios en el valor de *x*, partiendo de la situación más trivial *(x = 0)*, lo cual significa que las mejores soluciones sobrantes no se compartirán en lo absoluto. De esta manera, vemos cómo la metaheurística se degrada en una búsqueda paralela con *n* ramas (recordar que *n* hace referencia a la cantidad de soluciones iniciales) sin ningún tipo de *feedback* entre ellas. Si la atención es puesta en el espacio de búsqueda se verá cómo cada solución está confinada a un sector de tal espacio, sea bueno o malo, donde se inició. En este punto no se cuenta con una bandada sino con una cantidad determinada de aves volando en solitario donde ninguno se percata de la existencia del otro. En contraparte, si al moverse hacia el otro extremo *(x = k)* se trasladará todo el beneficio a las demás soluciones. Lo cual a primera vista puede parecer maravilloso, pero si se observa cómo repercute esto en el algoritmo se notará que optar por esta alternativa solo permitirá explorar el vecindario de soluciones respectivo al líder, ya que todas las demás soluciones estarían operando sobre él mismo, y todo el aporte que podrían realizar estos individuos se pierde. Volviendo al ejemplo natural es posible comparar está situación con una bandada de aves volando en hilera, dónde la totalidad del esfuerzo lo realiza el líder y por lo tanto se pierde el elemento sinérgico que le dio origen. A partir del análisis de los puntos extremos es posible ver cómo valores moderados de *x* supondrán una mejor performance del algoritmo.

Cuando el beneficio alcanza ambas ramas de la formación, y ya se crearon y evaluaron los vecindarios de soluciones asociados a cada individuo es necesario determinar el tiempo durante el cual el líder se mantendrá en esa posición antes de agotar sus fuerzas y necesitar un relevo. Esta situación está representada por el valor *m*, el cual indica la cantidad máxima de aleteos que realizará el líder antes de ser reemplazado por alguno de sus seguidores y ubicarse al final de una de ambas ramas de la formación. Es posible suponer que, si se toma valores pequeños para *m* significará que los intercambios en la posición del líder tendrán lugar con una frecuencia muy alta y el beneficio que los seguidores obtengan de él se verá reducido. En el algoritmo, esto puede interpretarse como que la diversificación de soluciones es menor ya que la solución líder tuvo poco tiempo para transferir sus mejores candidatos. En contraste, si al considerarse valores mayores para *m* se producirá un estancamiento ya que, una vez que cada solución individual converja en un sector del espacio de búsqueda las siguientes iteraciones solamente repetirán los resultados anteriores, aumentando el tiempo de procesamiento sin agregar plusvalía al resultado final.

2.4. Publicaciones relacionadas con MBO

En esta sección se realiza un análisis de las investigaciones publicadas en la literatura, a fin de que el lector posea una noción del camino recorrido con respecto al algoritmo MBO original.

En el estudio realizado por Quan-Ke Pan y Yan Dong [9] se aborda el problema de una línea de producción híbrida tomando el algoritmo desarrollado por Duman y añadiendo

algunas modificaciones a la versión original. Ellos utilizan un generador Glover para obtener la población inicial, y al momento de generar los vecindarios de soluciones hacen uso de los operadores de inserción e intercambio. Ambos operadores cuentan con la misma probabilidad de ser seleccionados para su posterior aplicación. Quizás la modificación más contrastante es el mecanismo de salto *(leaping mecanism)*. El objetivo de este mecanismo es evitar que la búsqueda se atasque en óptimos locales, lo cual se logra asociando una edad a cada solución generada. En el momento de su creación la edad se inicia con el valor "0", si una vez terminado el proceso de actualización no se encontró una mejor solución que reemplace la existente, entonces la edad de esa solución se incrementa en "1". Si dicha solución no es reemplazada por un mejor candidato después de varias iteraciones es posible que la búsqueda esté rondando un mínimo local. Entonces, lo que proponen en su investigación es tratar de reemplazarla por una nueva alternativa a fin de escapar de este óptimo local pero manteniendo algo de la información contenida en la solución actual. De esta forma, no se desecha todo el avance logrado a través de las iteraciones. Esto lo logran implementando conjuntamente un intercambio con una inserción, ya que el resultado obtenido no es fácilmente alcanzable aplicando sólo uno de los dos operadores. Para garantizar que la nueva solución sea lo más prometedora posible se genera todo un nuevo vecindario de soluciones y se toma la mejor, ésta será la que reemplace a la solución actual, las demás se descartan.

Ekrem Duman, Mitat Uysal y Ali Fuat Alkaya [10] por su parte realizan una comparación del clásico algoritmo propuesto por Duman con otras metaheurísticas existentes enfocado en el problema de asignación cuadrática y destacan la forma en la que los parámetros tales como, el número de soluciones (aves), el tamaño de los vecindarios de soluciones (velocidad de vuelo), la cantidad de iteraciones en las cuales la mejor solución encabezará la lista (el tiempo durante el cual el líder de la bandada cumplirá dicho rol) y el número de soluciones compartidas entre soluciones (el beneficio obtenido por las aves que siguen al líder) afectan el desempeño del algoritmo. De su análisis es posible realizar algunas conjeturas en cuanto a estos parámetros tales como que la función de performance en relación al número de soluciones forma una parábola invertida, es decir que el desempeño aumenta hasta alcanzar un número determinado de soluciones y luego decrece a medida que este número se incrementa. En cuanto al tamaño del vecindario generado por cada solución, puede decirse que el algoritmo se desempaña mejor cuando se trabaja con vecindarios reducidos. Esto puede explicarse mediante el hecho de que al aumentar el número de soluciones de un vecindario la búsqueda local realizada en él tiene que tratar con un mayor número de elementos y por tal se verá reflejado en la performance. Es decir que a pesar de que la búsqueda es más exhaustiva no aporta nuevos resultados. Por otro lado si la cantidad de iteraciones se mantiene en valores bajos afecta la velocidad de convergencia del algoritmo pero del mismo modo, según el análisis de los investigadores, queda demostrado que un número de iteraciones excesivas puede desencadenar que la búsqueda se atasque en un óptimo local. Por último evaluaron el efecto que tiene la cantidad de soluciones compartidas entre individuos adyacentes, según los estudios este valor debe ser bajo pero no puede ser "0" y tiene sentido ya que aumentar la cantidad de soluciones compartidas implica disminuir la cantidad de soluciones que cada elemento de la formación puede generar y esto reduce la exploración del espacio de búsqueda, sin embargo si cada elemento es libre de crear un vecindario completo de soluciones se estaría tratando con un problema de búsqueda en paralelo en el cual no existe relación alguna entre los elementos.

Vahit Tongur y Erkan Ülker [11] aplican el algoritmo original a los problemas de secuenciación en las líneas de producción remarcando las similitudes que este presenta con el algoritmo genético. Dentro de esas similitudes se hallan los operadores que toman partido en el algoritmo basado en la migración de las aves con los operadores de entrecruzamiento que se utilizan con algoritmos genéticos.

Al igual que el estudio mencionado en el párrafo anterior Ali Fuat Alkaya y Ramazan Algin [12] también aplican el algoritmo original pero contrastándolo con otras metaheurísticas como algoritmos genéticos, optimización basada en colonia de hormigas y *simulated annealing* proyectados al problema de neutralización de obstáculos.

Otro de los problemas en el cual fue testeado el desempeño de este algoritmo es en la esquematización de instalaciones en circuito cerrado donde el objetivo es reducir el costo del manejo de materiales mediante una óptima esquematización de las etapas productivas. La resolución a este interrogante puede verse en el trabajo realizado por Sadegh Niroomand, Abdollah Hadi-Vencheh, Ramazan Şahin y Béla Vizvári [13] en el cual el grupo de trabajo aporta una modificación al algoritmo original al incluir los operadores de mutación y entrecruzamiento utilizados en los algoritmos genéticos. Su propuesta es regenerar las soluciones compartidas por medio de estos operadores en lugar de compartirlas directamente.

Como lo han demostrado Eduardo Lalla-Ruiz, Christopher Expósito-Izquierdo, Yesica de Armas, Belén Melián-Batista, and J. Marcos Moreno-Vega [14] la optimización basada en la migración de las aves también mostró buenos resultados en la resolución de problemas tales como la organización de una terminal de contenedores o la programación de las actividades a realizar por las grúas de descarga ubicadas en los puertos. En su trabajo el equipo encara el problema centrándose en la capacidad del algoritmo de apuntar hacia las regiones del espacio de soluciones más prometedoras, esto lo logran utilizando un método de mejora propuesto por Lalla-Ruiz [15]. Su método consiste en que una vez obtenida una solución se la mejora por medio de un mecanismo de reinserción. A partir de esa solución mejorada se genera todo un nuevo vecindario por medio de un mecanismo de intercambio, luego se devuelve el mejor elemento del vecindario. Como puede verse en su trabajo, al emplear este método de mejora aumentó la velocidad de convergencia del algoritmo hacía mejores soluciones a costa de un leve incremento en el costo computacional.

Como puede apreciarse por el estado del arte aún queda mucho camino por recorrer y el futuro es prometedor en lo que respecta a esta nueva metaheurística, ya que ha demostrado tener muchos puntos de inflexión donde es posible introducir mejoras como así también es fácilmente combinable con alguna de las ya existentes.

Capítulo 3: Problema de ruteo vehicular

En las siguientes secciones puede apreciarse una introducción donde se explica el contexto en el cual se desarrolla el problema. Junto a esto se detalla la motivación asociada a la búsqueda de métodos de resolución óptimos. Seguidamente se hace mención de las investigaciones más relevantes que se han desarrollado sobre la materia, su surgimiento y la evolución de los métodos empleados. Para finalizar la sección se presenta la formulación del problema junto con el modelo matemático asociado.

3.1. Introducción

Cuando la humanidad comenzó a asentarse en lugares específicos formando las primeras civilizaciones tuvieron que enfrentar la necesidad de transportar los productos que no les era posible conseguir en su lugar de residencia. Ya sea porque no existía la posibilidad de fabricarlos o cultivarlos, o bien sólo se podía hacer en determinadas épocas del año. El problema fue encontrar la forma más práctica de transportar dichos recursos ya que algunos sólo eran aptos para el consumo durante periodos cortos de tiempo, como puede ser el caso de carnes u otros productos perecederos. Es aquí donde comienzan a surgir los primeros indicios de lo que actualmente se conoce como logística.

Más adelante en la historia de la humanidad y con el surgimiento del comercio existían personas dedicadas a surtir las urbes. Estos individuos acarreaban con una gran variedad de productos a cambio de una contribución monetaria. Es entonces cuando el esfuerzo logístico se orientó hacia el objetivo de reducir los costos de transporte a fin de ampliar el margen de ganancia.

En la época moderna la logística cumple un papel fundamental en las empresas. Su misión radica en facilitar la comunicación entre sucursales y la disponibilidad de los productos. En economías tan dinámicas, donde la ecuación costo-beneficio puede perder el balance en un instante, mantener un margen de ganancia significativo se vuelve una tarea ardua si no se cuenta con información rápida y actualizada. La necesidad de información hace que las personas se enfoquen en la búsqueda de mejores herramientas a modo de que les facilite y agilice la toma de decisiones. Con el advenimiento de la tecnología, incluyendo el desarrollo de modelos matemáticos y surgimiento de las computadoras, fue posible simplificar está tarea relegando el trabajo a las máquinas que hoy en día son tan imprescindibles para los negocios.

El primer modelo matemático adaptado al problema de logística de las empresas fue denominado problema de ruteo vehicular (*vehicle routing problem*) *o VRP* por sus siglas en inglés. Este problema fue inicialmente planteado y propuesto por G. Dantzig y J. Ramser [16] al final de la década de los ´50. Ellos establecieron la formulación matemática y el algoritmo para hallar la forma más económica de surtir un grupo de estaciones de servicio partiendo desde el depósito. Con el correr de los años este problema evolucionó hasta convertirse en un foco de investigación importante para profesionales de diferentes áreas. Tomó un gran protagonismo en el campo de los negocios y en otros emprendimientos tales como despliegues militares, sanitarios o de asistencia ante catástrofes. Esto fomentó que constantemente se estén implementando nuevos algoritmos para resolver instancias más

grandes y complejas de este modelo. Algunos de estos enfoques son exactos, como puede ser el método *Branch and Bound* [17] o *Branch and Cut* [18]. Lo cual implica que el resultado de aplicarlos siempre será una solución óptima. Pero debido a que estamos tratando con un problema del tipo *NP-hard*, su aplicación no proporcionará la solución en un tiempo razonable cuando se trabaje con instancias reales y grandes. Es por este motivo que surge otro enfoque orientado a la implementación de heurísticas o metaheurísticas. Como ejemplos de metaheurísticas adaptadas a este problema podemos mencionar *Tabu Search* [19], *Ant Colony Optimización* [20][21], Algoritmos Genéticos [22], entre otros. También, y como se dijo anteriormente, la incidencia de este problema en el campo empresarial alentó el surgimiento de una gran cantidad de variantes del algoritmo original. Es el caso del problema de ruteo vehicular con ventanas de tiempo o VRPTW, en donde a cada cliente se le asocia una ventana de tiempo para ser atendido. Otro caso es el problema de ruteo vehicular con depósitos múltiples o MDVRP que, como el nombre lo indica, posee múltiples puntos de partida y final (uno por cada depósito) para las diferentes rutas. También se encuentra el problema de ruteo vehicular dependiente por áreas o SDVRP, en el cual se estipula que los clientes pueden ser visitados por un subconjunto de camiones que a su vez pueden tener diferentes capacidades. Otro problema es el de ruteo vehicular abierto u OVRP, en donde no es necesario que los vehículos retornen al depósito. Por último se menciona el enfoque al cual está dirigido esta investigación llamado problema de ruteo vehicular con capacidad o CVRP, en el cual existe un depósito central desde el cual parten los camiones con el objetivo de recorrer la mayor cantidad posible de clientes minimizando la distancia total recorrida y considerando, como restricción principal, que cada cliente debe ser visitado por un único vehículo. Dentro de cada caso particular se tienen en cuenta objetivos diferentes. Mientras que otras configuraciones como MDVRP, SDVRP y CVRP buscan minimizar la distancia recorrida sin superar la capacidad de carga de los vehículos, el enfoque VRPTW y OVRP tienen como primera prioridad reducir la cantidad de vehículos utilizados y como segunda prioridad reducir la distancia recorrida.

3.2. Estado del arte

El problema del ruteo vehicular fue formulado originalmente por George Dantzig y Jhon Ramser en 1959 [16] como una generalización del problema del viajante de comercio. Ellos establecieron el modelo matemático y las primeras resoluciones utilizando los algoritmos existentes en la época. A medida que avanzó el tiempo, y cómo ya se dijo anteriormente, fue objeto de amplio estudio y se destinaron numerosas investigaciones que motivaron el surgimiento de diferentes variantes el problema cada una adaptada a distintos requerimientos.

Los primeros intentos en hallar soluciones óptimas para este problema condujeron a aplicar diferentes variantes de algoritmos exactos tales como *Branch and Cut*, *Branch and Bound* y posteriormente *Branch and Cut and Price* [23]. Estos algoritmos abordan la exploración del espacio de soluciones transformándolo en una estructura tipo árbol y realizan una búsqueda exhaustiva comparando todas las soluciones posibles con la mejor obtenida hasta el momento. Hacen uso del *backtraking* para recorrer dicho árbol, agregando la posibilidad de realizar también la poda de aquellas ramas del árbol sobre las que ya se tenga conocimiento de que no serán fructíferas. Además, a fin de reducir el tiempo computacional requerido para hallar una solución, se realizan algunas relajaciones en las restricciones

impuestas al problema a fin de ganar mayor flexibilidad. Un ejemplo de ello es la *Programación lineal de enteros (ILP)* [23]. Sin embargo el consumo de recursos por parte de estos métodos era demasiado y con el tiempo se fue indagando en mejores metodologías.

Con el avance de las investigaciones se puso el foco en los algoritmos heurísticos. Los cuales tomaban ciertas características del problema y permitían que la búsqueda ganara algo de experticia en lugar de dejar que el resultado sea producto de la exhaustividad. Debido a esto prometían una mejor performance que sus antecesores exactos. Dentro de la literatura puede verse el gran esfuerzo dedicado al estudio de estas técnicas sobre el problema subyacente. Como alguna de las heurísticas importantes puede mencionarse *cluster-first, route-second* [24], la cual primero ubica *m* soluciones semillas y construye un clúster de clientes para cada semilla, teniendo en cuenta que la suma de las distancias entre clientes debe satisfacer la restricción de capacidad. Otros investigadores han propuestos heurísticas para realizar la búsqueda en un vecindario de soluciones (como es el caso de [25]) en un contexto dinámico dnde situaciones como el retiro o entrega de bienes ocurren en tiempo real. Seguidamente varios grupos de investigadores han propuesto numerosas heurísticas adaptadas a distintos enfoques según las necesidades tenidas en consideración. En el trabajo realizado por Kumar y Panneerselvam [26] se realiza un recorrido más amplio sobre este aspecto.

El correr de los años también capitalizó conocimiento en la materia y permitió avanzar un escalón más en lo que a métodos de resolución se refiere. Es aquí cuando surge una variedad nueva de algoritmos que traen consigo grandes ventajas, una de ellas era su independencia del problema. A diferencia de las heurísticas, está independencia le permitía adaptarse a varios problemas sin que guarden relación entre si y por otro lado estaba el buen aprovechamiento de los recursos que denotaban. Fue así como las metaheurísticas se convirtieron quizás en el campo más explorado por la comunidad científica en los últimos años.

En [27] el equipo presenta una metaheurística llamada *Guided Tabu Search* en dónde se combina una búsqueda tabú con una búsqueda local guiada. Tal estrategia es aplicada a una de las variantes del problema original denominada *problema de ruteo vehicular con dos restricciones de carga dimensionales o 2L-CVRP*. Su esfuerzo está dirigido a estudiar la situación en la que los bienes a ser dispuestos en los camiones no pueden ser apilados y únicamente pueden esparcirse sobre la superficie de carga. En su propuesta la exploración del espacio de búsqueda es realizada por una búsqueda tabú. La cual es alentada por un mecanismo de exploración denominado *búsqueda local guiada* y cumple el objetivo aumentando el grado de diversificación de las soluciones. Además cuentan con un paquete de heurísticas que permiten realizar una exploración inteligente del espacio de búsqueda. Los objetivos del equipo fueron puestos en dos configuraciones del mismo problema. La secuencial – esto es cuando la carga y descarga de los productos se hace de forma que guarda una correlatividad con el orden en que se visitan los clientes – o sin restricciones – a diferencia del caso anterior esta correlatividad se pierde –.

En [26] también se realiza un sumario de algunas de las metaheurísticas utilizadas para resolver, en este caso, el *problema de ruteo vehicular con ventana de tiempo*. En el documento ellos refieren el trabajo realizado por Blanton y Wainwright [28] cómo los primeros en aplicar algoritmos genéticos para la resolución de este problema. En este enfoque

se utiliza una metaheurística híbrida ya que combina algoritmos genéticos con heurísticas *greedy*. De esta forma mientras que el algoritmo genético se enfoca en la búsqueda de un ordenamiento de los clientes, la heurística *greedy* se encarga de la construcción de una solución factible. Aunque, según las fuentes, el equipo se adjudica ser los precursores en la implementación de los algoritmos genéticos sobre este problema muchos otros investigadores siguieron su iniciativa. Lo que produjo el surgimiento de distintos enfoques a mencionar el caso de *algoritmos genéticos con heurísticas constructivas* [28, 29], *búsqueda local* [30, 31] y otras metaheurísticas como *búsqueda tabú* [32] y *optimización por colonia de hormigas* [33]. En otro trabajo relacionado a la misma variante del problema [34] los investigadores incorporan una flota de vehículos limitada, lo cual aporta un mayor realismo a los problemas logísticos. Ellos proveen un límite máximo a los recursos utilizados y muestran que la *búsqueda tabú* se acerca bastante a ese límite. Por otro lado este algoritmo cuenta con cierta estabilidad y también aporta buenos resultados para la versión estándar del VRPTW. Una especificación de la versión con ventana de tiempo es la que abordan los investigadores en [35]. En este caso se trata del *problema de ruteo vehicular con ventana de tiempo periódico (PVRPTW)* en el cual el lapso de planificación se extiende a varios días y cada cliente debe ser abastecido en una ventana de tiempo específica. Para resolverlo utilizaron una *mejora en la optimización por colonia de hormigas (IACO)* en donde hacen uso de una matriz de feromonas multidimensional. El objetivo es acumular información heurística respecto a varios días, anexo a esto se incorporó el operador de *cross-over* doble para mejorar el desempeño del algoritmo.

En [21] se ataca otra variante del problema llamada DVRC *(problema de ruteo vehicular dinámico)* y se propone un *sistema basado en colonia de hormigas (ACS)* para su resolución. La forma en que ellos encaran el problema es dividiendo el día de trabajo en espacios de tiempo individuales. De esta manera se genera una secuencia de VRP estáticos donde cada uno de ellos es resuelto por ACS. Este sistema de optimización también permite transferir buenas soluciones entre divisiones de tiempo. Agregado a esto, definieron un nuevo conjunto de instancias públicas a fin de poder testear los resultados de su desarrollo. Y luego de un estudio computacional logró demostrarse que el algoritmo provee buenos resultados tanto en problemas artificiales como reales.

Para la variante del *problema de ruteo vehicular con capacidad (CVRP)* los autores de [36] proponen un *algoritmo genético celular (cGA)*. El cual es un tipo de metaheurística descentralizada basada en población la cual presenta una alta performance en términos de calidad de soluciones y número de funciones de evaluación.

En el artículo [37] proponen otra variación del mismo problema denominada *problema de ruteo vehicular con depósitos múltiples y depósitos múltiples entre rutas (MDVRP-IDR)* el cual es una especificación del mismo problema con depósitos múltiples. En su trabajo ellos proponen una heurística combinando el principio de memoria adaptativa, una *búsqueda tabú* para la solución de los subproblemas y *programación de enteros*.

Como puede verse por el estado del arte, si bien éste es un problema muy estudiado en términos generales, aún queda mucho por recorrer en algunas de sus variaciones. En este trabajo se pone el acento en el *problema de ruteo vehicular con capacidad* del cual no se encontró mucha documentación debidamente avalada y ninguna documentación que lo plantee utilizando el *MBO*, por tal motivo esta investigación tiene por objetivo, entre otros,

sentar precedente de la utilización de dicho algoritmo en la resolución del problema planteado y utilizarlo como punto de partida para futuros proyectos.

3.3. Formulación del problema

Como ya se ha mencionado en este documento encararemos la versión con capacidad del problema de ruteo vehicular o *CVRP* y ha llegado el momento de introducir el modelo matemático que respalda el trabajo realizado.

El problema se ha asociado a un conjunto de restricciones en donde la que cobra una mayor relevancia es aquella vinculada con la capacidad, ya que determina la viabilidad de la solución obtenida.

Inicialmente el CVRP está definido como un grafo completo y no dirigido *G = (V, E)*, donde *V* representa el conjunto de vértices y cada vértice representa a un cliente. Cada cliente posee una demanda no negativa asociada y se hace uso de un valor como identificador, reservando el valor "0" para el depósito. A su vez el grafo está conectado por un conjunto de arcos *E* en donde cada arco tiene un peso asociado que representa la distancia recorrida entre el cliente *i* y el cliente *j*. Una flota de tamaño fijo K conformada por vehículos con capacidad Q se encuentra en el depósito a la espera de realizar el recorrido asignado durante el cual se visitarán a todos los clientes.

A continuación se presentan las variables que intervienen en el problema:

- *G = {V, E},*
- $V = \{v_0, \dots, v_n\}$ donde v_0 representa al depósito y $v_1, \dots, v_n$ representan a los clientes,
- c_i representa la demanda *q* para el cliente *i* con $i \in V$,
- d_{ij} representa la distancia entre el cliente *i* y el cliente *j*,
- $K = \{k_1, \dots, k_m\}$ representa la flota de vehículos,
- $Q = \{q_0, \dots, q_m\}$ representa la capacidad de cada vehículo (en caso de que la flota sea homogénea) $q_i = q_j \; con \; 0 \leq i, j \leq m$,

El orden en el que se visitan los clientes queda representado por:

- $x_{ij}^k \in \{0,1\}$ *: x = 1* si el cliente *j* es visitado directamente del cliente *i* por el vehículo *k*
- $y_i^k \in \{0,1\}$ *: y = 1* si el cliente *i* está en el recorrido del vehículo *k*

A su vez cada una de las variables está sujeta a las siguientes restricciones: (i) cada cliente debe pertenecer sólo a una ruta determinada, (ii) cada ruta comienza y termina en el depósito y (iii) la demanda total de todos los clientes visitados no debe exceder la capacidad *Q* del vehículo. (iv) El objetivo es encontrar la configuración de rutas que permita visitar a la totalidad de los clientes recorriendo la menor distancia. Lo cual queda representado por la siguiente función objetivo:

$$Min \sum_{k=0}^{m} \sum_{i=0}^{n} \sum_{j=0}^{n} d_{ij} x_{ij}^k \, , con \; i \neq j$$

Es posible representar las restricciones impuestas anteriormente según el siguiente modelo matemático:

$$\sum_{k=1}^{m}\sum_{j=1}^{n} x_{ij}^{k} = K, para\ i = 0 \quad (1)$$

$$\sum_{j=1}^{n} x_{ij}^{k} = 1,\ \ para\ i = 0\ y\ k \in \{1, \dots, m\} \quad (2)$$

$$\sum_{j=1}^{n} x_{ji}^{k} = 1,\ \ para\ i = 0\ y\ k \in \{1, \dots, m\} \quad (3)$$

$$\sum_{k=1}^{m}\sum_{j=0}^{n} x_{ij}^{k} = 1, \forall i\ \in \{1, \dots, n\}\ y\ con\ i \neq j \quad (4)$$

$$\sum_{k=1}^{m}\sum_{j=0}^{n} x_{ij}^{k} = 1, \forall i\ \in \{1, \dots, n\}\ y\ con\ i \neq j \quad (5)$$

$$\sum_{i=1}^{n} c_i \times \sum_{j=0}^{n} x_{ij}^{k} \leq q_k, para\ k \in \{1, \dots, m\} \quad (6)$$

La restricción (1) impide que la cantidad de rutas sea mayor que la cantidad de camiones indicando con el signo ≤ que puede esta puede ser menor.

Las restricciones (2) y (3) especifican respectivamente que para cada vehículo de la flota existirá un único arco saliente del depósito que tendrá su contraparte entrante. Lo cual aplicándolo en conjunto se traduce en que la cantidad de arcos salientes y entrantes tiene su correspondencia en el tamaño de la flota y garantiza que cada vehículo realiza un tour completo.

De forma similar las restricciones (4) y (5) determinan que para cada nodo del grafo existirá un único arco entrante y un único arco saliente lo cual garantiza que cada cliente será visitado una única vez por un único vehículo.

Por último la restricción (6) controla que para cada vehículo la demanda total asociada a la ruta sea menor o igual que la capacidad del camión designado.

Capítulo 4: Optimización basada en la migración de las aves aplicada al problema de ruteo vehicular con capacidad

En este capítulo se describen los detalles propios del algoritmo aplicado al problema de ruteo vehicular. En una primera instancia se comenta sobre la representación adoptada para la solución y la forma de generar el vecindario de soluciones. Seguido a esto, se realiza una breve explicación de cómo se inicializa la población. Para finalizar se confecciona un detalle sobre el método que se adoptó para explorar el espacio de búsqueda.

4.1. Representación de la solución

Para expresar la solución en el contexto del problema adoptado se utilizó un vector de enteros que representa una permutación sin repetición. En la cual cada elemento del mismo puede entenderse como un índice. De esta forma, los valores comprendidos entre el valor *1* y *N* representan a cada uno de los clientes que deben ser visitados, y los valores comprendidos entre *N+1* y *M* representan a los camiones destinados a realizar cada uno de los recorridos. Puede notarse que se dejó excluido el número 0 ya que está reservado para denotar el depósito o punto de partida de la flota. A fin de esclarecer la interpretación entiéndase que la sucesión de valores asociados a los clientes, ubicados luego de un valor mayor a *N*, corresponden a la ruta asignada al camión representado por el índice respectivo. Por ejemplo, si contamos con una cantidad de 10 clientes y 4 camiones para recorrerlos, en donde cada camión es asignado únicamente a una ruta específica, un posible vector solución quedaría representado como se muestra en la Figura 04. Donde puede verse cómo el camión 11 (o el camión asociado a la ruta 11) recorre los clientes 4, 8 y 2, el camión 12 hace lo mismo para los clientes 3, 6 y 9, el camión 14 no tiene clientes asignados y finalmente el camión 13 visita a los clientes 10, 5, 7 y 1.

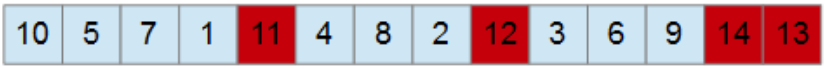

Figura 04: Representación de la solución.

La ventaja de esta representación radica en que muchas de las restricciones propias del problema se encuentran implícitamente representadas. Este es el caso de que un cliente únicamente puede pertenecer al recorrido realizado por un camión, al ser un vector que no admite repetición en sus elementos esto queda garantizado de antemano. De la misma forma ocurre cuando se solicita que el recorrido llevado a cabo por cada camión debe comenzar y finalizar en el depósito. Cada vez que se encuentra un índice que escape al valor máximo de clientes dado se tiene en cuenta que el recorrido de ese vehículo terminó y se calcula la distancia al depósito desde el último punto visitado.

4.2. Estructura de la bandada

Como se comentó en la sección 2.2 en la optimización basada en la migración de las aves las soluciones se agrupan cual aves en una bandada y su comportamiento se asemeja al de estos animales. Por tal motivo al iniciar la ejecución del algoritmo el primer paso es establecer los integrantes de la bandada y la forma en que estos estarán organizados. De esta manera contamos con una solución líder, no necesariamente la mejor, y con dos listas enlazadas que representarán a las ramas izquierda y derecha de la formación. El líder será el encargado de encabezar la búsqueda y facilitar el trabajo a sus seguidores inmediatos, estos a su vez, harán lo mismo con los suyos y así sucesivamente hasta llegar al final de la formación. Al mismo tiempo cada solución irá realizando una exploración del terreno de soluciones a medida que el vuelo se desarrolla. Un esquema típico de una formación de soluciones en "V" puede apreciarse en la Figura 05.

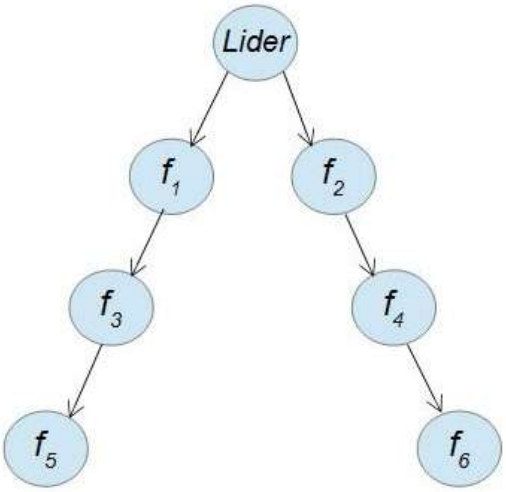

Figura 05: Estructura de la bandada.

4.3. Evaluación inicial

Cada una de las soluciones generadas en la etapa anterior es evaluada inmediatamente después de su generación a fin de determinar su valor de fitness. Una vez fijado éste, se procede a la optimización de las soluciones, la cual se lleva a cabo mediante la generación de una cantidad determinada de nuevas soluciones. Éstas conformarán el vecindario para la solución dada, explicado en la siguiente subsección. La etapa de evaluación, además de determinar el valor de fitness que le corresponde a cada solución, cuenta con un sistema de penalización mediante el cual se degradan las soluciones que no cumplen las restricciones dadas, en este caso particular se trata de la restricción de capacidad. Esa degradación tiene por objetivo descartar de manera prioritaria aquellas soluciones no aptas a fin de que otras, probablemente con un fitness mayor pero que satisfacen la restricción establecida, se mantengan en juego para dirigir la búsqueda en las siguientes iteraciones. La forma en que se produce este efecto es aumentando el fitness de la solución penalizada en un factor *p* multiplicado por la cantidad de veces que se superó la capacidad de un vehículo en un recorrido dado. Cómo lo muestra la siguiente ecuación:

$$P = p \times \sum (h_i^k \times y_i^k)$$

Donde,

- P es el valor total de la penalización,
- p determina el factor de penalización,
- $h_i^k \in \{0,1\}$: *h = 1* si la demanda del cliente *i* supera la capacidad del camion *k,*
- $y_i^k \in \{0,1\}$ *: y = 1* si el cliente *i* está en el recorrido del camión *k*

De esta forma, el fitness se verá incrementado sustancialmente provocando que esa solución ya no signifique una mejora y por lo tanto no sea seleccionada para futuras exploraciones. Este proceso se repetirá con cada nueva solución generada y permite que a lo largo del tiempo permanezcan sólo las soluciones factibles, concentrando la exploración en los sectores del espacio de búsqueda que conducirán a mejores soluciones.

4.4. Estructura del vecindario

Cada solución de la bandada tiene un vecindario de soluciones asociado, el cual es generado a partir de dicha solución. Este vecindario comprende el mecanismo de exploración mediante el cual se irá avanzando por el espacio de búsqueda a fin de lograr la convergencia hacia mejores sectores, y por lo tanto, a mejores soluciones. Cabe hacer una salvedad entre la estructura que tendrá el vecindario de la solución líder con el de sus seguidores. Esto es así a causa de que el primer elemento es el encargado de encabezar la búsqueda y debido a eso afrontar el esfuerzo asociado a tal tarea. Es por esta razón que, la cantidad de soluciones generadas por el líder de la bandada alcanzará completamente el valor *k* y favorecerá a sus seguidores inmediatos con las *x* mejores soluciones, y el resto será descartado. De esta manera, los demás elementos verán aliviada su misión de exploración en el sentido de que ya no será necesario alcanzar el valor *k* de nuevas soluciones sino que, en su lugar, deberá alcanzar un valor *k-x* porque ya cuenta con los *x* elementos cedidos por la solución antecesora. En el caso de aquellas soluciones que se encuentren al final de la bandada verán que no les es posible entregar ningún beneficio, por la obvia razón de que ellas constituyen el punto final de la formación. En este caso, todas las soluciones no utilizadas serán destruidas y el ciclo comenzará nuevamente desde la posición líder.

En la Figura 06 puede observarse a una bandada hipotética acompañada de sus respectivos vecindarios de soluciones (denotados como $s_1, s_2, s_3, s_4, \dots, s_k$ y $s_1, s_2, s_3, s_4, \dots, s_{k-x}$ para el caso del líder y de sus seguidores respectivamente) y el beneficio en cuestión (denotado como $b_1, \dots, b_x$).

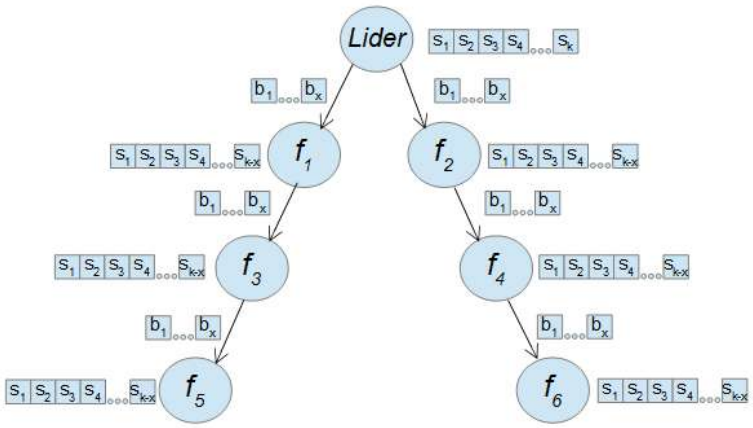

Figura 06: Estructura del vecindario.

Cada solución generada durante la ejecución del algoritmo se logra por medio de alguno de los tres operadores descriptos a continuación.

4.4.1. Operador de inserción

El objetivo de este operador es efectuar el movimiento de un elemento del vector solución a una posición diferente de la actual en el mismo vector. Por lo cual primero se procede, por medio de una selección aleatoria, a determinar el elemento que será afectado por el operador y la posición a la cual será trasladado para luego aplicar la operación en sí. En otras palabras un cliente de la posición ***i*** es removido de la ruta asociada al camión ***k*** y reinsertado en una nueva posición ***i'*** correspondiente a una ruta ***k'*** asignada a otro camión o dentro de la misma ruta ***k*** en cualquiera de las posiciones posibles, como se muestra en la Figura 07.

El mismo desempeña un buen trabajo de exploración ya que permite alcanzar soluciones que difícilmente se obtengan con otros operadores. Esto es debido a sus características altamente disruptivas. Por lo que su aplicación, si bien es útil, debe hacerse con ciertos resguardos para evitar que los cambios de foco en el espacio de búsqueda se den con demasiada frecuencia y por lo tanto impidan focalizar la búsqueda en un área prometedora.

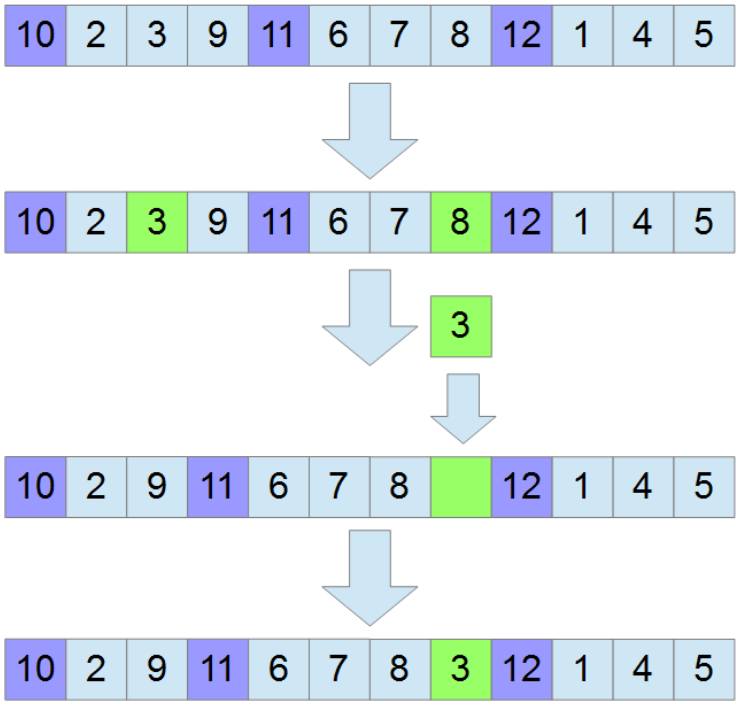

Figura 07: Operador de inserción.

4.4.2. Operador de inversión

Este operador tiene como finalidad realizar una alteración de la solución ocasionando que una porción del vector en cuestión resulte invertida. Para esto primero es necesario seleccionar las posiciones que delimitarán la sección a invertir y luego poder aplicar la operación misma. Para expresarlo en un modo más formal podemos decir que seleccionado un cliente de la posición ***i*** y otro de la posición ***i'*** (no necesariamente perteneciente a un mismo recorrido) se invierten todos los elementos comprendidos entre las posiciones dadas independientemente de la ruta a la cual estén asignados, como se muestra en la Figura 08.

Quizás sea el operador principal en el problema abordado. Posee buen desempeño tanto para la exploración como para la explotación de sectores particulares, pero tiende a converger con mucha rapidez y provoca el estancamiento. Por tal, debe ser asistido por alguno de los otros dos operadores.

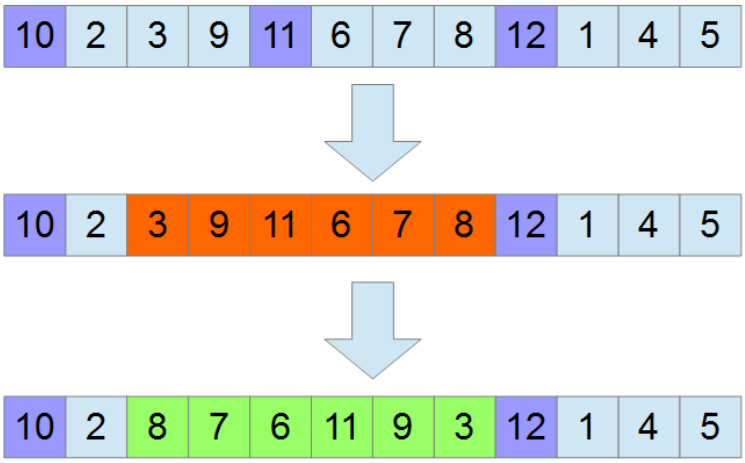

Figura 08: Operador de inversión.

4.4.3. Operador de intercambio

Este operador desempeña una función similar a la realizada por los dos operadores anteriores. Al igual que en sus precedentes primero se procede a seleccionar los elementos del vector que serán intercambiados para luego aplicar el cambio en sí. En otras palabras, una vez seleccionados el elemento del vector solución ubicado en la posición ***i*** y el elemento ubicado en la posición ***i'*** estos se intercambian tal como puede apreciarse en la Figura 09.

Es importante resaltar que cualquier elemento del vector puede ser seleccionado para el intercambio (tanto clientes o camiones), esto apunta a generar una mayor diversidad en las soluciones y por lo tanto ampliar los sectores explorados del espacio de búsqueda.

Esta variante posee características de comportamiento muy similares a las que se comentaron para el operador de inserción. Demuestra un buen desempeño para la exploración y en combinación con inserciones alcanzan muy buenos resultados, tanto que en varios de los documentos consultados [2][9][13] es la opción utilizada para generar nuevas soluciones a partir de las semillas. Sin embargo, al aplicarlo en el problema de ruteo vehicular con capacidad requiere que se combine con el operador de inversión para explotar los puntos críticos.

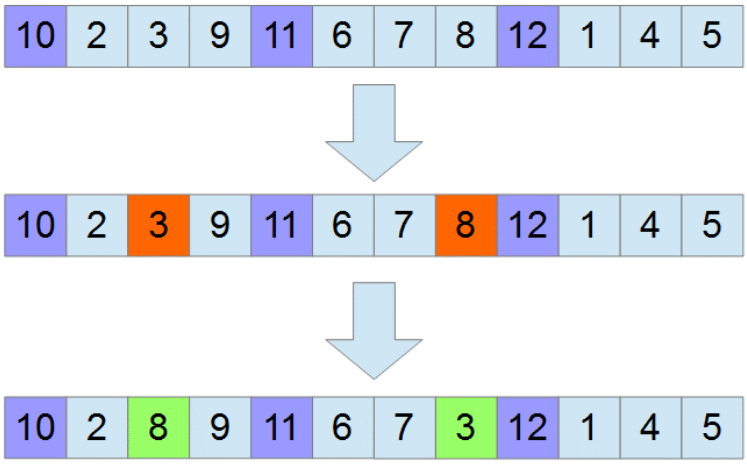

Figura 09: Operador de intercambio.

4.5. Inicialización de la población

En un primer paso, al comenzar la ejecución del algoritmo, se inicializa la población creando las soluciones que compondrán a la bandada. Esto se logra por medio de un generador que construye el vector de enteros con índices que comienzan en el valor 1 y finalizan en el valor $N+M$, donde N y M corresponden con la cantidad de clientes y la cantidad de rutas respectivamente. Una vez que se cuenta con el vector inicializado se procede a aplicar una serie de permutaciones para dotar de cierta heterogeneidad a la solución. Esto se logra aplicando el operador de intercambio una cantidad $M+N$ de veces sobre el vector obtenido anteriormente. De esta forma se procede con el resto de las soluciones iniciales hasta completar la totalidad de la bandada.

4.6. Exploración del espacio de búsqueda

El trabajo de exploración está asociado a los tres operadores mencionados en la sección 4.4 (inserción, inversión e intercambio) y cada uno de ellos tiene un efecto único en la forma de recorrer el espacio de búsqueda. Algunos tienen la característica de dotar a las nuevas soluciones de un alto nivel de diversidad, en otras palabras es posible catalogarlos como operadores muy disruptivos y tenerlos en mente como una herramienta muy útil para la exploración del espacio de soluciones. Esto es así porque rompen con la secuencia de búsqueda actual y encauzan los esfuerzos hacia nuevas secciones del espacio. Lo cual es de gran utilidad al momento de querer escapar de óptimos locales. Por otro lado se cuenta con operadores que demostraron tener un gran nivel de convergencia y su mayor potencial se obtiene al explotar sectores prometedores del espacio debido que, como resultado, producen soluciones muy similares a la solución inicial que reciben. Esto implica que las aproximaciones en la búsqueda son más sutiles.

Cada uno de estos operadores cuenta con algunas desventajas. En el caso de los operadores muy disruptivos, en donde podemos agrupar al operador de intercambio y de inserción, al generar soluciones muy alejadas de la solución semilla los resultados pueden ser

poco satisfactorios. Esto sucede ya que el algoritmo necesita mayor esfuerzo computacional para guiar la búsqueda a sectores prometedores. Cabe destacar que este hecho resulta más evidente cuando únicamente se avanza mediante inserciones. Por otro lado cuando sólo se emplea el operador de intercambio es posible notar una leve mejora en la calidad de las soluciones obtenidas pero el resultado todavía se mantiene alejado de los límites aceptables. Ahora, si se dirige la atención hacia el operador de inversión, que favorece la explotación, es posible ver que esta característica provoca que la búsqueda se centre en un único punto prometedor del espacio, como consecuencia se obtienen resultados claramente mejores, aunque sin llegar a ser valores óptimos. Esto puede verse ejemplificado por la Tabla 02 en la sección 5.2.

Antes de que se introduzca el mecanismo de exploración adoptado es necesario remarcar que se consideraron varias opciones. Algunas propuestas por la bibliografía y que de hecho resultaron útiles para los problemas abordados por los autores de tales documentos. Pero que en el caso del problema de ruteo vehicular demostraron tener un pobre desempeño brindando soluciones muy alejadas de los valores aceptables. Por otro lado se consideró la utilización conjunta, no simultánea, de pares de operadores. Aquí se tomaron todas las combinaciones de estos tres operadores.

Claramente puede verse en la Tabla 02 que las mejores combinaciones de operadores fueron aquellas que involucran al operador de inversión. Resaltando la combinación de inversión e inserción por alcanzar los mejores resultados tanto en valores promedios como también en puntos mínimos. Sin embargo no se debería descartar completamente el aporte del operador de intercambio al momento de contribuir con la exploración del espacio de búsqueda. En conclusión según las características del problema se puede asumir, por el momento, que el operador que tiene mayor incidencia en la calidad del resultado es el operador de inversión, seguido por el de intercambio y finalmente el de inserción, esto quedará comprobado en la siguiente sección. Y la comparación puede verse en la Tabla 02 en la sección 5.2.

Por tal motivo buscamos un mecanismo de exploración que permitiera la utilización conjunta mas no simultanea de los tres operadores, con el objetivo de aprovechar en mayor medida todos los recursos disponibles. Tal búsqueda concluyó en un mecanismo de selección aleatoria, el cual permite que cada nueva solución generada sea el resultado de aplicar una de estas tres alternativas operatorias. La ventaja de esto se encuentra en que es posible realizar una explotación eficiente gracias al operador de inversión y al mismo tiempo dotar de cierta diversidad (exploración) a las soluciones por medio de los operadores de inserción e intercambio.

4.7. Intercambio del líder

Una vez alcanzada la distancia *m* llega el momento de que el líder actual obtenga su relevo. De esta manera él pasa a una ubicación más privilegiada mientras otra ave, con más energía, encabeza la búsqueda. El nuevo líder cumplirá su rol durante *m* nuevas iteraciones antes de que otra ave lo releve.

En el modelo realizado, estos intercambios se realizan con una simple actualización de punteros. Tal herramienta brinda la abstracción necesaria para mantener una bandada de soluciones en formación independientemente de cómo se encuentren almacenadas físicamente.

Una vez alcanzado el momento de realizar el intercambio, el líder actual se posiciona al final de la rama que proporcionará al nuevo líder, mientras que quién antes era su seguidor pasa a desempeñar el rol de liderazgo. El ciclo se repite cada vez que se alcanza el valor *m* alternado, en cada caso, la rama que proporciona al nuevo candidato. El procedimiento queda ilustrada por la Figura 10.

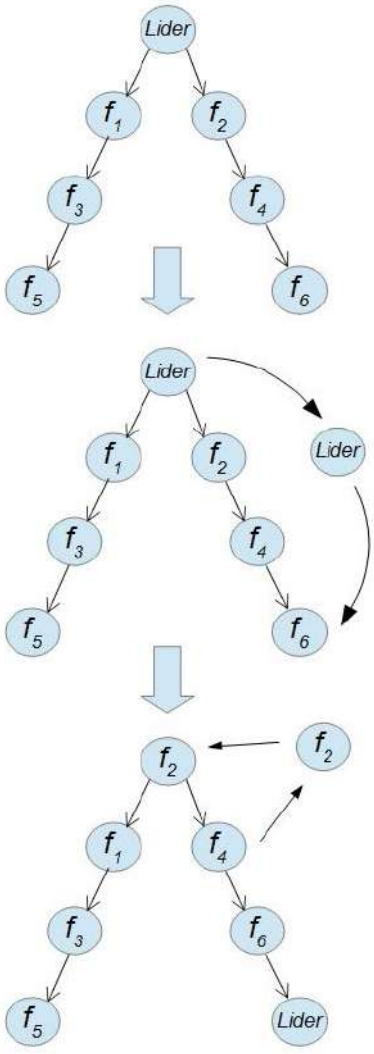

Figura 10: Intercambio del líder.

Capítulo 5: Experimentación y análisis de resultados

En este capítulo se describen los experimentos realizados y se analizan los resultados obtenidos al solucionar CVRP usando MBO. Inicialmente, se introducen el ambiente de ejecución sobre el que se llevó a cabo y las instancias utilizadas. Seguidamente se explica la selección de los parámetros que permiten obtener el mayor beneficio del algoritmo. Finalmente, se presentan los resultados y su análisis, el cual se realiza desde el punto de vista de la calidad y del desempeño del algoritmo.

5.1. Diseño Experimental y analítico

El algoritmo MBO, para resolver CVRP, fue implementado en el lenguaje de programación C++ por ser un lenguaje de bajo nivel que permite un manejo de memoria dinámica y la utilización del paradigma orientado a objetos. Este manejo de memoria dinámica permite reducir el costo computacional durante la ejecución del algoritmo. En tanto que, la utilización de un paradigma orientado a objetos posibilita un análisis y diseño del problema más intuitivo. El modelo del software puede verse en la Figura 11.

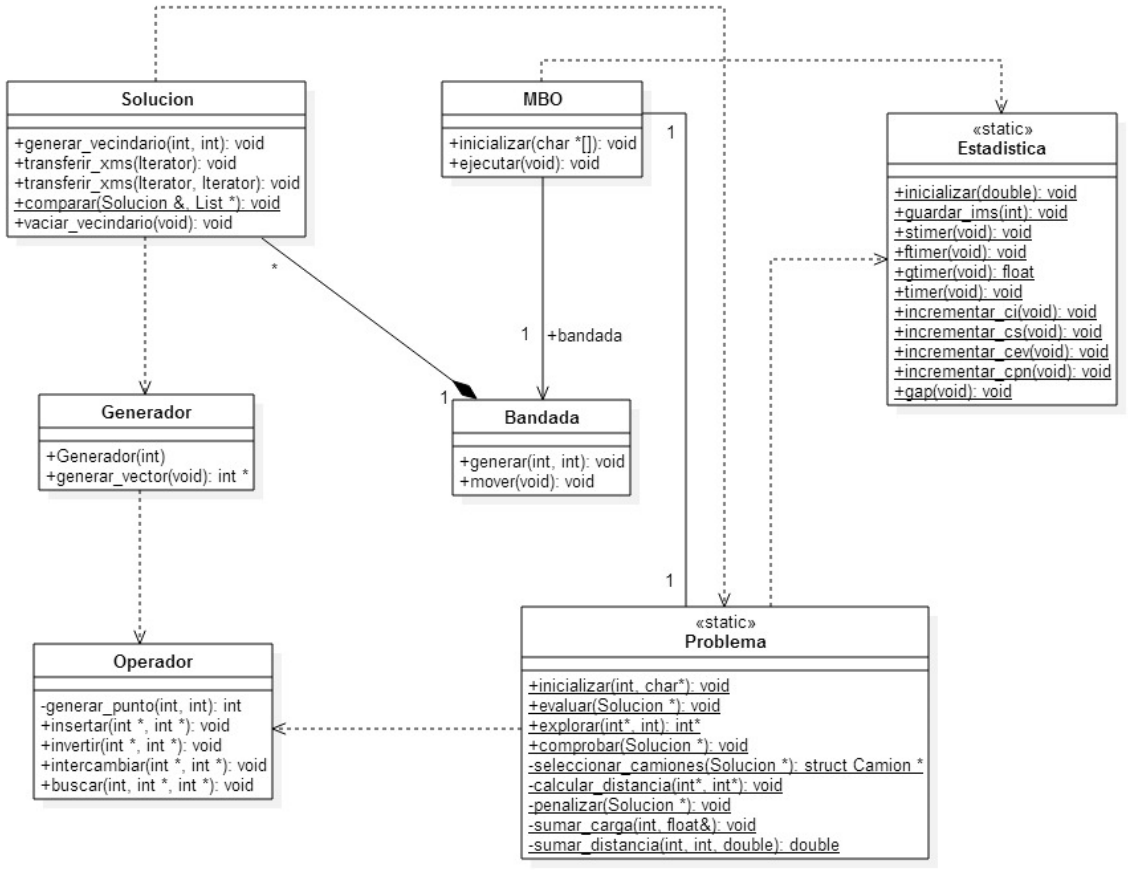

Figura 11: Modelo estático del algoritmo MBO.

Para llevar a cabo la experimentación se recurrió a las instancias publicadas por Rochat y Taillard [38]. Las cuales cuentan con carteras de clientes variadas en cantidades de 75, 100 y 150 puntos de atención respectivamente. Las cuales siguen un esquema donde el depósito se ubica en el centro del plano y se encuentra rodeado por clústeres de clientes. Cada uno con un valor de demanda asociado, el cual fue tomado de una distribución exponencial, según lo mencionado en [39]. El conjunto utilizado se conforma de 12 instancias diferentes agrupadas según la cantidad de clientes en 3 subgrupos de cuatro instancias. En la Figura 12 puede verse un ejemplo de la instancia tai75a, la cual cuenta con el depósito resaltado en color blanco ubicado en el origen de coordenadas y los clientes (puntos negros) esparcidos a su alrededor.

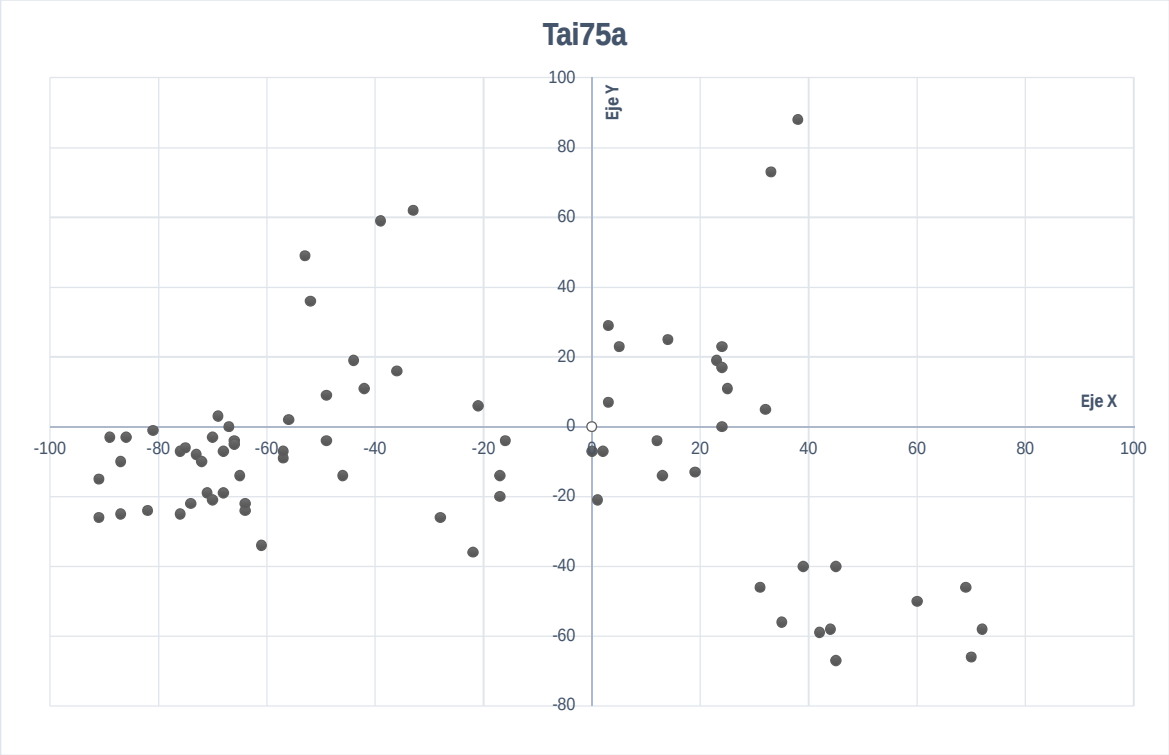

Figura 12: Distribución de clientes para la instancia tai75a.

Para la configuración de los parámetros de entrada del algoritmo se realizó una experimentación y un análisis de resultados, previos a la experimentación final, que se detalla en la sección 5.2.

Para la experimentación final y debido a la naturaleza estocástica del algoritmo se realizaron 30 ejecuciones independientes para cada instancia. En total se llevaron a cabo 360 ejecuciones (12 instancias x 30 ejecuciones cada una), en un clúster de PC’s con un procesador Intel Core i7 a 3,90 GHz y 4GB de RAM.

Con respecto al procedimiento empleado para el análisis de los resultados se realizó un estudio del comportamiento del algoritmo MBO, considerando un GAP normalizado entre la mejor solución hallada para cada combinación de parámetros (*n, m, k, x)* y el óptimo conocido para la instancia de prueba. El cual se calcula teniendo en cuenta el valor de fitness evaluado y el valor óptimo conocido, como lo muestra la siguiente ecuación:

$$GAP = \frac{fitness - \acute{o}ptimo}{\acute{o}ptimo} \times 100$$

Este valor determinará la calidad (fitness) de una solución dada, considerando ésta como mejor cuanto más se aproxime al valor cero.

Un resumen de resultados puede apreciarse en la Tabla 03. Dicho análisis consiste en evaluar valores extremos y medios para *x* con el objetivo de agilizar la experimentación, y agregar nuevos valores sólo si los resultados lo indican. Para el análisis se clasificaron las ejecuciones según los distintos valores de *n*, realizando una segunda clasificación según los valores de *m*. De cada subgrupo se extrajo la combinación (*n, m, k, x*) que aportó la mejor aproximación.

5.2. Configuración paramétrica

Para hallar la configuración de los parámetros de entrada adecuada se ejecutó el algoritmo bajo 375 combinaciones diferentes de valores paramétricos sobre la instancia tai75b. Estas 375 variantes surgen de combinar los valores paramétricos que se muestran en la Tabla 01 propuesta por Duman et. al en [6]. Por cada combinación de parámetros se realizaron 10 ejecuciones, totalizando unas 3750 ejecuciones. Cabe mencionar que se elige la instancia *tai75b* ya que se encuentra documentada con el detalle suficiente para permitir un buen análisis de desempeño.

Parámetro	Valores
Número de aves en la bandada *(n)*	7, 13, 25, 51, 101
Velocidad de vuelo *(k)*	3, 7, 15, 25, 51, 101
Distancia recorrida por el líder *(m)*	5, 10, 20, 30, 40
WTS *(x)*	*min, avg, max*

Tabla 01: Valores de configuración del algoritmo.

Para determinar la configuración paramétrica de MBO es necesario tener en cuenta ciertas restricciones. A mencionar, en el caso del número de soluciones en la bandada, determinado por la variable *n*, es recomendable seleccionar un valor impar, de tal forma que la estructura quede conformada por un líder en el frente y dos ramas de igual longitud a los extremos. Además, existe una correspondencia entre el tamaño del vecindario generado por cada solución (*k*) y la cantidad de soluciones que es compartida por ella (*x*). El motivo de tal relación radica en que, para el líder, esta cantidad asciende a *2x* por los dos seguidores que se beneficiarán de él, a diferencia de los demás que sólo comparten *x* soluciones, también es necesario contar con una solución más (la mejor) para ser intercambiada por la solución que dio origen al vecindario. En conclusión el valor que tomará la variable *k* debe ser al menos *2x+1*. En el caso de las variables *m* y *K* no poseen restricciones particulares salvo las impuestas por el hardware sobre el que se ejecutará el algoritmo. En la Tabla 03 es posible tomar noción de los resultados aportados por la experimentación. Este resumen de los datos obtenidos se generó considerando el mejor resultado obtenido para cada combinación de

parámetros. Los valores resaltados en negrita corresponden a la combinación de valores que aportó la mejor aproximación del total de ejecuciones.

Por otro lado también es necesario considerar la interacción y el desempeño de los tres operadores utilizados, como se mencionó en la sección 4.6. A fin de justificar tal análisis en la Tabla 02 se expresan los resultados de aplicar cada uno de ellos de forma individual y combinada, respectivamente.

Operador/Combinación	Mejor GAP
Inserción	36,7996%
Inversión	0,5275%
Intercambio	11,7356%
Inserción + inversión	0,0995%
Intercambio + inversión	1,2397%
Intercambio + inserción	1,1570%

Tabla 02: Resultados de explorar el espacio de búsqueda con distintas combinaciones de operadores

n	m	k	x	GAP min
7	5	25	6	0,8220%
7	10	51	1	1,3854%
7	20	3	1	3,0135%
7	30	51	12	1,8679%
7	40	7	3	1,2869%
13	5	7	3	1,3161%
13	10	15	1	0,2889%
13	20	25	1	0,0781%
13	30	51	25	0,0491%
13	40	101	1	0,1145%
25	5	7	1	0,6298%
25	10	51	25	0,0935%
25	20	25	1	0,0491%
25	30	101	1	0,0643%
25	40	51	12	0,1838%
51	5	101	1	0,0491%
51	10	15	1	0,1795%
51	20	7	1	0,0935%
51	30	101	1	0,0261%
51	40	25	1	**0,0179%**
101	5	51	1	0,0646%
101	10	25	1	0,0491%
101	20	25	1	0,1235%
101	30	3	1	0,0646%
101	40	7	1	0,0646%

Tabla 03: Valores mínimos de GAP para cada configuración paramétrica.

A fin de profundizar la experimentación se consideraron otros valores de *x* para la combinación de *n, m,* y *k* seleccionada, los resultados pueden apreciarse en la Tabla 04.

x	GAP promedio	GAP mínimo
3	2,6003%	0,4810%
6	1,8318%	0,6881%
9	3,8915%	0,4979%

Tabla 04: Valores mínimos de GAP para otros valores de *x*.

Con el fin de realizar un análisis completo de los resultados obtenidos es necesario hacer hincapié en el rol que, cada uno de los parámetros de entrada, cumple en el desempeño del algoritmo. Por tal motivo en las siguientes líneas se explicará brevemente la función de cada uno.

El valor de *n* es quizás el más importante de todo el algoritmo MBO, debido a que determina la cantidad de puntos de partida que tendrá la exploración, los cuales se corresponden con cada solución en la bandada. Y conjuntamente con *m* fija la profundidad que tendrá la exploración en sí. Como ya se mencionó este valor no puede ser arbitrariamente grande como tampoco muy pequeño. Esto se debe a que mientras más aumente el número de individuos en la formación más rápidamente se llegará a la condición de terminación. Aunque dicho límite puede ampliarse a gusto, esa decisión implica un aumento en el consumo de los recursos y del tiempo de ejecución. Mientras que, si tomamos un valor pequeño el proceso de búsqueda será pobre. Entonces, queda claro que el valor que tome *n* debe alcanzar el equilibrio justo entre la diversidad de la búsqueda y los recursos requeridos. Es posible notar el hecho al ver la Tabla 03 ya que sin importar el valor de *m* los mejores resultados ocurrieron cuando la formación está compuesta por 51 individuos *(n=51)*.

La Tabla 03 también da cuenta de la estrecha relación que mantiene el tamaño de la bandada y la distancia de vuelo del líder *(n y m)*. Es posible observar (en las filas 1 y 5), que cuando se cuenta con una bandada reducida, la performance empeora a medida que el líder se mantiene por más tiempo en su rol. La permanencia prolongada implica que el algoritmo se atasque en un óptimo local ante la falta de nuevas alternativas para la búsqueda. Del mismo modo si vemos en las filas 16 y 20 (Tabla 03), para bandadas más grandes, a medida que la permanencia del líder aumenta mejor es la performance. Esto corrobora el hecho de que el tiempo durante el cual el líder se mantenga como tal es proporcional a la profundidad alcanzada en la exploración del espacio de búsqueda.

Una situación similar ocurre con el valor de *k*. En la naturaleza este valor representa el poder inducido de cada ave en la bandada. En nuestro modelo representa la cantidad de soluciones que cada individuo de la bandada generará, es decir, establece el radio de búsqueda que tendrá la exploración a lo largo de la ejecución. El valor de *k* que resultará en mejores resultados dependerá estrechamente del valor *n*. Para dejarlo más claro, si trabajamos con bandadas reducidas, el área del espacio de búsqueda que debe barrer cada solución tendrá que ser mayor, por tal motivo en esa configuración esperaremos ver valores más grandes de *k* (esto puede comprobarse comparando las filas 1 y 3 de la Tabla 03); en cambio, en un escenario inverso, al tener una gran cantidad de aves en la bandada cada una de ellas tendrá

que abarcar un área menor (como lo muestran las filas 16 y 20 de la Tabla 03). El problema que se presenta si el valor del parámetro en cuestión no sigue estos lineamientos será que, en el primer caso, quedarán zonas del espacio sin explorar con el riesgo de que la mejor solución permanezca inalcanzable en dichos sectores. En el escenario alterno, si se selecciona un *k* muy grande para un valor de *n* grande se producirá un solapamiento de los radios de búsqueda y esto ocasionará una degradación del rendimiento ya que se recorrerá zonas del espacio visitadas previamente por otras soluciones.

Parte de ese poder inducido, mencionado en el párrafo anterior, es utilizado por su seguidor directo para evitar el agotamiento prematuro. Ese papel lo desempeña la variable *x*. Ahora bien, desde el punto de vista de las metaheurísticas, este valor puede entenderse como el grado de diversidad que una solución aporta a la siguiente. En otras palabras el vecindario de soluciones que genere cada ave en la bandada excepto el líder recibirá una cantidad *x* de soluciones de su vecino antecedente. Entonces, su propio vecindario se verá acotado en esa misma cantidad. Según *x* sea mayor o menor el vecindario que una solución sea capaz de generar disminuirá o aumentará respectivamente.

Finalmente y basado en los datos obtenidos se comprobó que a medida que la bandada se incrementa mayor es la profundidad que alcanzará la búsqueda y este valor está ligado a la distancia de vuelo. Por otra parte, el tamaño del vecindario generado por cada una de las soluciones demostró un mejor comportamiento para valores moderados como se suponía que suceda. En cuanto al parámetro *x*, que representa el beneficio obtenido entre los integrantes, los mejores resultados se obtuvieron con valores bajos aunque puede sufrir unos leves incrementos cuando se trabaja con valores pequeños de *k*. Para concluir se seleccionó la configuración $n = 51$, $m = 40$, $k = 25$, $x = 1$, ya que fue la que mejor performance logró. En cuanto al valor *K*, está claro que mientras mayor sea el espacio de búsqueda mejores serán los resultados obtenidos y que las limitantes de este parámetro son únicamente temporales o en un segundo nivel físicas, es decir, *K* puede tomar cualquier valor arbitrariamente grande siempre que el tiempo consumido por el algoritmo y los atributos físicos del equipo en donde se ejecute lo permitan. Siendo estas dos condiciones una limitante a la hora de seleccionar su valor. Por lo expuesto se determinó un valor de *K=2.000.000* para la etapa de experimentación con el objetivo que se recorra una cantidad de soluciones suficiente para que el algoritmo tenga una convergencia aceptable y luego, en las pruebas finales, elevarlo a *K=8.000.000* lo cual amplía la cantidad de soluciones evaluadas significativamente. En total, con el valor de *K* elegido para la ejecución final, se realizaron 164 iteraciones para cada ejecución individual.

5.3. Análisis de resultados

En esta sección se realiza un análisis de los resultados obtenidos en la experimentación desde dos perspectivas. En la primera subsección se aborda el análisis desde el punto de vista de la calidad de las soluciones obtenidas. Mientras que, en la segunda subsección, el foco se centrará en el desempeño.

5.3.1. Análisis de calidad

En los siguientes párrafos se presentan los resultados obtenidos seguidos de un breve análisis de los datos. En la Tabla 05 se presenta el GAP mínimo y la mejor solución obtenida por MBO para cada instancia, así como también la solución óptima. Mientras que en la Figura 13 se muestra el promedio de los mejores valores de GAP para cada grupo de instancias.

Instancia	GAP	Mejor Solución	Solución óptima
tai75a.dat	0,7505%	1630,51	1618,36
tai75b.dat	0,0311%	1345,06	1344,64
tai75c.dat	1,0328%	1304,34	1291,01
tai75d.dat	0,7999%	1376,34	1365,42
tai100a.dat	2,4871%	2092,11	2041,34
tai100b.dat	0,6319%	1952,87	1940,61
tai100c.dat	1,2325%	1423,53	1406,20
tai100d.dat	1,4935%	1604,87	1581,25
tai150a.dat	2,1245%	3120,14	3055,23
ai150b.dat	1,3296%	2764,04	2727,77
tai150c.dat	2,5842%	2402,36	2341,84
tai150d.dat	2,7252%	2717,48	2645,39

Tabla 05: Mejores GAP's obtenidos por MBO para cada instancia CVRP evaluada.

A partir del análisis de la Tabla 05 y de la Figura 13 se infiere que, en el caso de las instancias compuestas de 75 clientes (las de menor tamaño) es posible notar que todos los resultados tuvieron una buena aproximación al óptimo. Entre ellos se destacan los aportados por la instancia *tai75b*, los cuales registraron la mejor calidad en cuanto a las soluciones obtenidas, alejándose del valor óptimo sólo un 0,0311%. Los menos favorables del conjunto fueron los obtenidos para la instancia *tai75c*, éstos se alejaron un 1,0328%. Y en promedio los resultados provistos por las primeras cuatro instancias mostraron un valor superior al óptimo por un 0,7752%. Como es posible notar MBO demostró un comportamiento más que aceptable en la evaluación de las instancias de menor tamaño ya que los resultados obtenidos no son considerablemente mayores que el óptimo.

A medida que el tamaño de las instancias se incrementa queda a la vista como también se incrementa la dificultad de alcanzar valores óptimos, pero aun así se mantienen muy buenos resultados. Para el caso de las instancias medianas (del orden de 100 clientes) se destacan los resultados provistos para la instancia *tai100b* los cuales alcanzan un GAP mínimo de 0,6319%. Queda reflejado que éste es un resultado muy cercano al valor óptimo conocido para esa instancia. Mientras que para esta categoría el peor rendimiento se reflejó en los resultados obtenidos para la instancia *tai100a*, los cuales alcanzaron un GAP de 2,4871%. Sin embargo, a pesar de ser este el peor resultado para el grupo de 100 clientes, aún se mantiene muy cercano al óptimo. Y en promedio las soluciones obtenidas a partir de

la evaluación de las instancias de este grupo se encuentran en el orden del 1,3630% por encima del óptimo.

En el caso de las instancias de 150 clientes, los resultados obtenidos mostraron un incremento respecto de las categorías anteriores pero nuevamente no se alejaron significativamente de los GAP's óptimos. En el caso más favorable, la instancia *tai150b*, el resultado estuvo un 1,3296% por encima del valor óptimo. Como puede verse en la Tabla 05 este GAP se posiciona mejor que los obtenidos para las instancias *tai100a* y *tai100d*. En contraparte el peor desempeño lo lleva la instancia *tai150d*, la cual produjo un resultado 2,7252 % mayor que el óptimo conocido pero sin embargo levemente superior que el resultado obtenido para la instancia *tai100a*. Haciendo un balance general, el desempeño de MBO para este rango de clientes se posicionó los resultados un 2,3544 % por encima del óptimo en valores promedio.

La similitud de los GAP's obtenidos a partir de los resultados en los diferentes grupos de instancias permite establecer que MBO es un algoritmo que puede escalar eficientemente al incrementar el tamaño de las mismas. En la Figura 13 se muestra el aumento de los mejores valores de GAP promedios obtenidos para cada grupo de instancias.

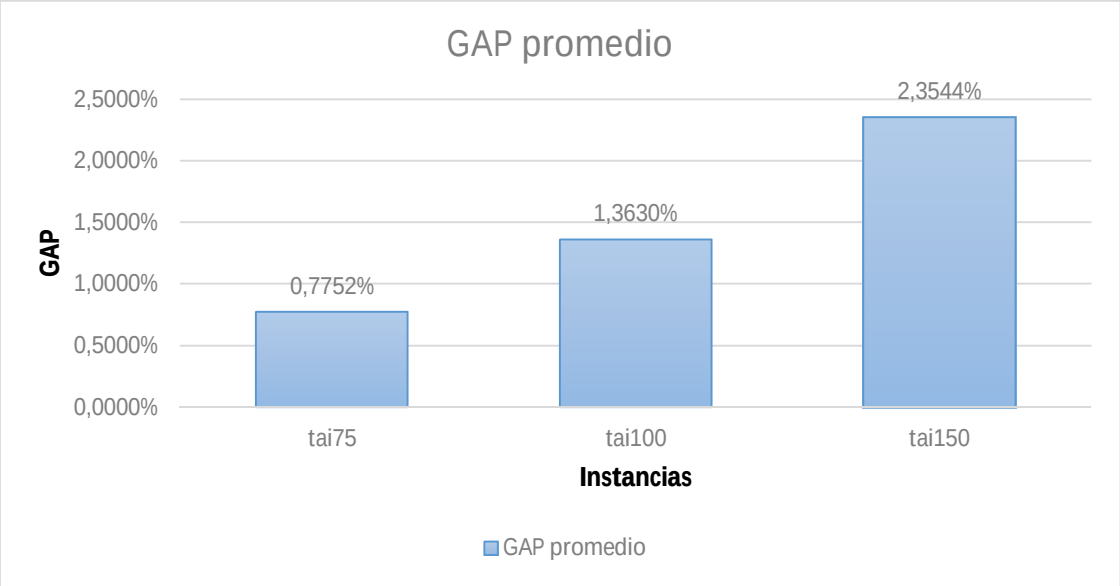

Figura 13: Evolución de la calidad de soluciones en valores de GAP promedio.

Como parte del estudio también se realizó una categorización de los resultados obtenidos según la calidad de las soluciones. De acuerdo a esta directriz se conformaron cuatro grupos de soluciones clasificadas según su valor de GAP en: mayores que un 4%, entre un 4% y un 3%, entre un 3% y un 2% y menores al 1%. En la Tabla 06 puede apreciarse la clasificación mencionada.

Instancia	GAP <= 1%	1% < GAP <= 2%	2% < GAP <= 3%	3% < GAP
tai75a	1	5	7	17
tai75b	16	6	4	4
tai75c	0	3	1	26
tai75d	1	6	21	2
tai100a	0	0	6	24
tai100b	1	6	11	12
tai100c	0	17	6	7
tai100d	0	3	3	24
tai150a	0	0	1	29
tai150b	0	1	2	27
tai150c	0	0	1	29
tai150d	0	0	3	27

Tabla 06: Categorización de los resultados.

Esta clasificación permite ver con claridad la distribución de los resultados obtenidos, y marca la calidad de las soluciones obtenidas según la complejidad de las distintas instancias. Para el caso del grupo con 75 clientes se puede ver, en general, la buena calidad de éstas. El 59,17% de los resultados arrojados por las ejecuciones hechas en el primer grupo se ubica por debajo del 3%, mientras que el 40,83% lo hace por encima de este valor (ver Figura 14). De todas las instancias pertenecientes a esta categoría se destaca la instancia *tai75b* con el 53.33% de los resultados obtenidos por debajo del 1%. En contraparte el peor lugar se lo llevan los resultados que brinda la instancia *tai75c*. Para este caso el grueso de las soluciones obtenidas (86,67%) logró un GAP por encima del 3%.

Para el grupo de instancias medianas, del orden de los 100 clientes, se puede apreciar como los resultados provistos por el total de ejecuciones se distribuye casi equitativamente según el GAP de las soluciones sea mayor o menor al 3%. Ya que casi la mitad de éstos (44.17%) posee un GAP menor al 3% (ver Figura 15). Dentro de este grupo de instancias es posible notar que se destacan las soluciones obtenidas para la instancia *tai100b,* ya que son las únicas que lograron valores de GAP por debajo del 1%. Éstas representan el 3,34% de las soluciones totales obtenidas para esta instancia. Mientras tanto, otra instancia que merece mención es la *tai100c*. En este caso una amplia mayoría de las soluciones obtenidas, el 76,67%, se ubica con un GAP menor al 3%. Y junto con la instancia *tai100b* son las que aportan las mejores soluciones de su grupo.

Finalmente, en el caso del último grupo de instancias con 150 clientes, se puede apreciar que el 6,67% de las soluciones pudieron superar la barrera del 3% mientras que el 93.33% de éstas se mantuvo por sobre este límite. En este grupo destaca la instancia *tai150b* con el 3,34% de las soluciones ubicadas en el rango que va entre el 2% y el 1% y el 6,67% entre los límites del 3% y el 2%. El peor resultado lo obtuvieron las instancias *tai150a* y *tai150c* en las cuales sólo el 3,37% de las soluciones pudieron sobrepasar la barrera del 3%. A fin de llevar a cabo un estudio de los resultados más detallado y acorde a la complejidad de las instancias resulta necesario incrementar los intervalos de categorización de los valores de GAP. Por tal motivo en la Tabla 07 se emplea una nueva categorización para el caso particular de los resultados obtenidos a partir de las instancias con 150 clientes. En dicha

tabla se clasifican las soluciones según su GAP en: mayores al 6%, entre un 6% y un 5%, entre un 5% y un 4% y menores al 4%.

Instancia	GAP <= 4%	4% < GAP <= 5%	5% < GAP <= 6%	6% < GAP
tai150a	2	3	5	20
tai150b	6	6	6	12
tai150c	2	10	4	12
tai150d	8	3	8	9

Tabla 07: Categorización expandida para los resultados del grupo tai150.

Al tener un panorama que aporta mayor claridad en la distribución de las soluciones obtenidas para este grupo en particular, es posible notar la buena calidad de los resultados. En general, para todo el grupo, el 52,5% de los resultados logró posicionarse por debajo del 6% mientras que el 47,5% no pudo superar este límite (ver Figura 16). Ahora si comparamos los resultados en el contexto de este grupo pueden verse que las soluciones provistas por las cuatro instancias mostraron un desempeño muy similar. Todas alcanzaron porcentajes similares por debajo de 6%. Sin embargo se destacan las obtenidas a partir de la instancia *tai150d* con el 63,34% de las soluciones por debajo de este umbral y un 26,64% de éstas por debajo del 4%. Los resultados de las instancias *tai150b* y *tai150c* quedaron muy cerca de los aportados por *tai150d* con un porcentaje, por debajo del umbral del 6%, que corresponde al 60% y 53,34% respectivamente. La menos favorecida del conjunto fue la instancia *tai150a*. La cual solo logró que el 33.34% de sus soluciones tuvieran un GAP menor al 6%.

En este breve análisis puede dar cuenta de los buenos resultados que MBO brinda al trabajar con instancias de baja, media y alta complejidad. Haciendo un balance general el 88,95% de las soluciones se posicionaron con un error menor al 6%. Esto vuelve a MBO una estrategia prometedora y motiva a continuar las investigaciones con nuevas estrategias de búsqueda para reducir este margen.

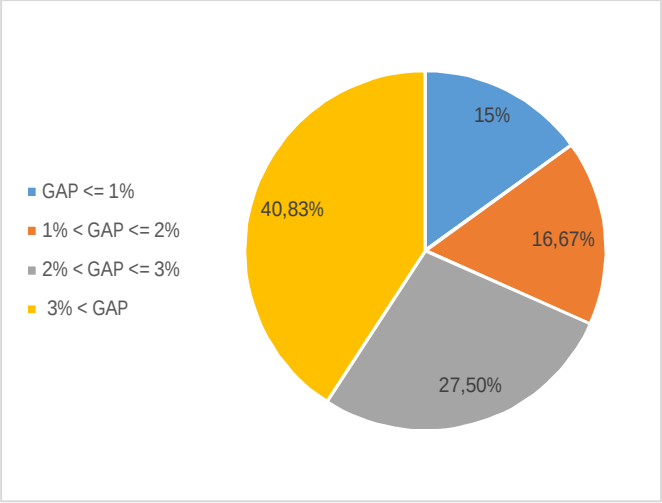

Figura 14: Categorización de resultados obtenidos por MBO para el grupo de instancias con 75 clientes según su GAP.

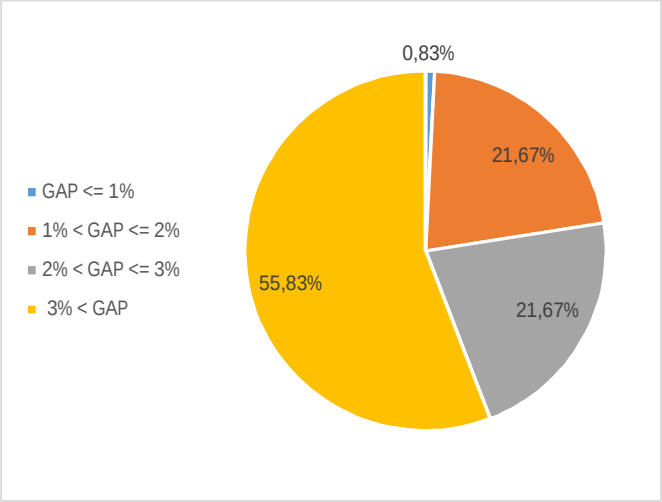

Figura 15: Categorización de resultados obtenidos por MBO para el grupo de instancias con 100 clientes según su GAP.

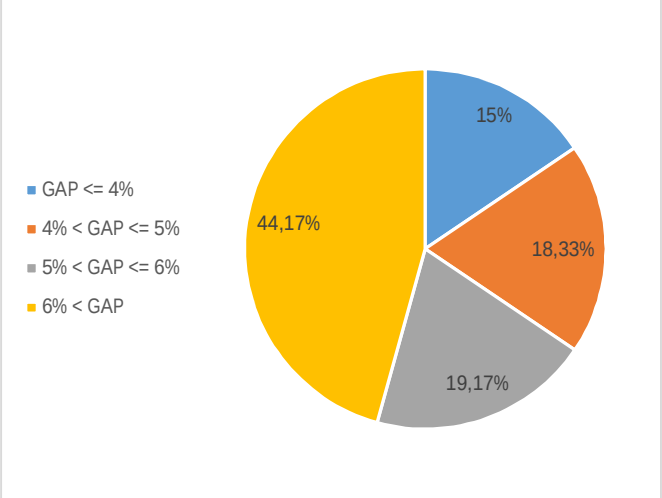

Figura 16: Categorización de resultados obtenidos por MBO para el grupo de instancias con 150 clientes según su GAP.

5.3.2. Análisis de desempeño

En la sección anterior quedaron expresados los buenos resultados logrados por esta primera implementación de MBO. El panorama mejora si se consideran los tiempos necesarios para hallar tales resultados. Por esto en las próximas líneas se llevará a cabo un análisis del desempeño de la estrategia. En la Tabla 08 puede verse el detalle del tiempo promedio requerido por MBO para obtener la mejor solución y el promedio del tiempo total de ejecución, además de la iteración promedio donde la mejor solución fue hallada.

Instancia	Tiempo mejor solución (seg.)	Tiempo total promedio	Iteracion mejor solucion
tai75a.dat	5,82	19,45	49
tai75b.dat	6,55	19,48	57
tai75c.dat	**3,32**	**19,32**	29
tai75d.dat	3,77	19,415	32
tai100a.dat	11,68	24	83
tai100b.dat	8,75	23,925	62
tai100c.dat	16,05	23,925	112
tai100d.dat	**8,07**	**23,835**	58
tai150a.dat	**26,97**	32,895	137
tai150b.dat	32,67	33,1	163
tai150c.dat	29,82	32,8	151
tai150d.dat	32,71	**32,765**	163

Tabla 08: Iteración y tiempo en que se halló la mejor solución junto con el tiempo total en segundos.

Comenzando por el grupo con menor cantidad de clientes, puede verse que fue posible para MBO hallar buenas soluciones en un tiempo mínimo. Lo cual deja clara la gran velocidad que posee está estrategia. En promedio, le tomó 4,80 segundos converger a la mejor solución. Mientras tanto, para finalizar la ejecución demoró 19,43 segundos (en promedio).

Con respecto a las instancias con 100 clientes, si bien el tiempo de procesamiento aumentó, no lo hizo en cantidades significativas. Con solo unos segundos de diferencia este grupo de instancias se procesó en un promedio de 23,93 segundos. Aunque no ocurrió lo mismo con el tiempo requerido para la convergencia. Para este grupo, a MBO, le tomó 10,22 segundos en encontrar la mejor solución, siempre hablando en valores promedio.

Finalmente, en las instancias con 150 clientes, puede verse que tanto el tiempo necesario para el procesamiento como el requerido para hallar la mejor solución prácticamente fueron iguales en magnitud. Al margen de esto es necesario remarcar nuevamente la velocidad de procesamiento con la que se encuentra esta solución. Al algoritmo le tomó un promedio de 32,85 segundos procesar y converger a la mejor solución. Este es un tiempo más que competitivo y como se pudo ver en la sección 5.2.1 los resultados alcanzados son de una muy buena calidad. Esto permite canalizar futuras investigaciones en la búsqueda de métodos que posibiliten mejorar la calidad de las soluciones encontradas. El escenario completo puede verse en la Figura 17 al comparar los tiempos de ejecución totales requeridos por cada instancia.

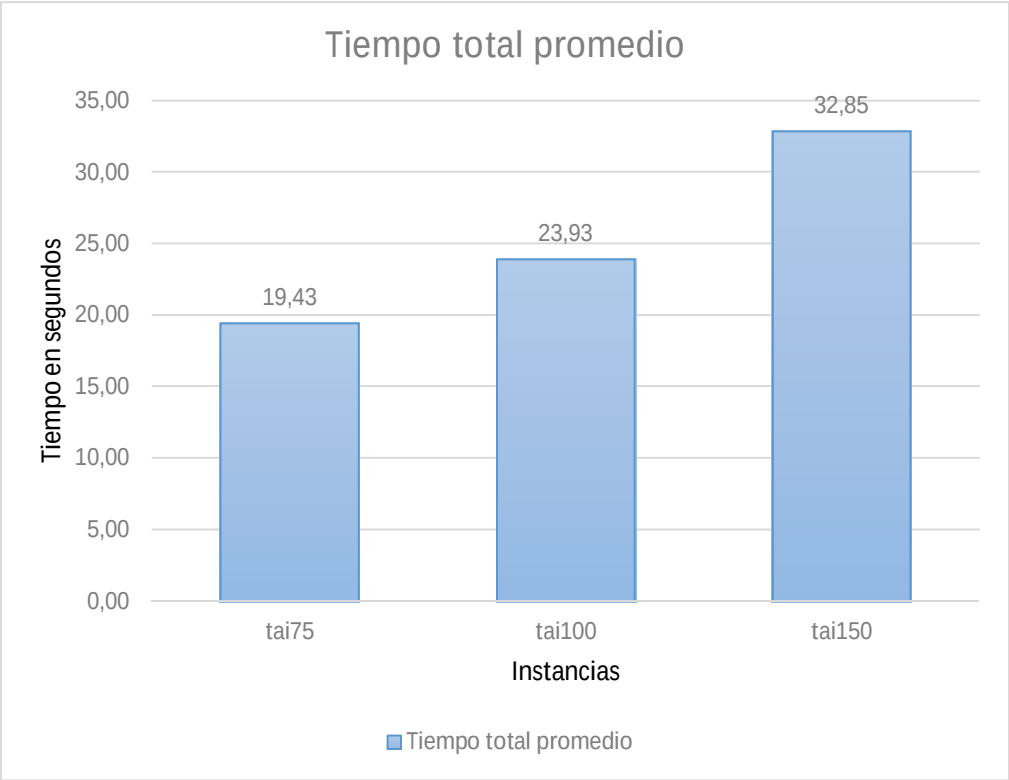

Figura 17: Comparativa de los tiempos de ejecución totales.

Capítulo 6: Conclusiones

A lo largo de este documento se mostraron las bases de esta nueva metaheurística llamada *Migration Birds Optimization (MBO)*, que nace del mecanismo de migración de las aves. La misma aporta al área una herramienta útil para la resolución de problemas complejos gracias a su gran poder de exploración y explotación del espacio de soluciones. La cantidad de parámetros de entrada que definen su desempeño implica un gran desafío a la hora de hallar la combinación correcta de valores. Para ello se ha experimentado teniendo en cuenta un abanico amplio de éstos con el fin de abarcar todas las combinaciones posibles. La combinación elegida permitió alcanzar soluciones de muy buena calidad. Y posibilitó comprobar las suposiciones que fueron planteadas en la sección 2.3 y que Duman ya había establecido en su investigación.

El desempeño de MBO fue puesto a prueba con el problema de ruteo vehicular con capacidad. Esta variante ya contaba con una reconocida dificultad en su resolución que no resultó ajena a las pruebas llevadas a cabo en el transcurso de esta investigación. Para ello se desarrolló una exhaustiva experimentación a fin de determinar la parametrización óptima del algoritmo para después pasar a la experimentación final. Esta se centró en un total de 12 instancias, propuestas por Taillard, compuestas por tres grupos de cuatro instancias cada una con 75, 100 y 150 clientes. Asociado a esto, se implementó una técnica de exploración basada en la selección aleatoria de tres operadores: inversión, intercambio e inserción. Los cuales mostraron un comportamiento diferente y como consecuencia aportaron capacidades tanto de exploración y explotación del espacio de búsqueda. Partiendo de las instancias de menor tamaño se logró obtener soluciones de una calidad competitiva en un tiempo mínimo. Cualidad que prácticamente se mantuvo en las instancias medianas y grandes. Ya que en la mayoría de los casos el desfasaje de las soluciones obtenidas con respecto a la óptima fue menor al 3%.

Junto a la calidad de los resultados obtenidos, cabe recalcar la gran capacidad de exploración y explotación que posee está técnica. Tales características quedaron en evidencia al comparar el tiempo de ejecución requerido con el tamaño del espacio de búsqueda (determinado por la cantidad de clientes) explorado, ya que en el peor de los casos rondó los 30 segundos. Obteniéndose valores cercanos al óptimo con una gran rapidez. En las instancias chicas y medianas, esto puede notarse en el margen significativo que existe entre el tiempo que se tarda en encontrar la mejor solución y el tiempo total de ejecución. Mientras que en las instancias de mayor tamaño ambos tiempos no reflejan una diferencia sustancial, los valores aportados por éstos se mantienen bajos. Todo lo expuesto anteriormente permite afirmar que MBO es un algoritmo eficiente que posee una gran tolerancia al aumento del tamaño de las instancias.

Finalmente, cabe resaltar una vez más que los resultados obtenidos han sido de una calidad aceptable para un primer enfoque de esta nueva técnica a un problema tan desafiante como es el ruteo vehicular con capacidad, y mencionar las grandes posibilidades que presenta esta técnica gracias al poco tiempo de procesamiento que insume. Queda para desarrollos futuros trabajar en la mejora del método de exploración para así alcanzar el óptimo en la mayor cantidad de instancias posibles. Contando con la ventaja de que gracias a los tiempos obtenidos se dispone de un margen de acción que permite llevar a cabo esta tarea y mantener

valores de tiempo competitivos. Dentro de las posibilidades, para futuros desarrollos, cabe la hibridación del mismo con otras técnicas de aproximación existentes que actualmente cuentan con un buen desempeño.

Como aprendizaje personal, encarar este trabajo cultivó mi experiencia en varias áreas. Por un lado me permitió poner a prueba los conocimientos adquiridos en todos estos años. Como así también dar los primeros pasos en la camino de investigador, sumado a la posibilidad de aplicar una nueva técnica con todos los desafíos que esto implica. Tal fue el caso de pulir los conocimientos adquiridos en las últimas materias de la carrea de Ingeniería en Sistemas como rescatar viejos saberes. Obstáculo al que me enfrenté al programar en C++, un leguaje que aprendí en los primeros años de carrera. Esta experiencia también me permitió potenciar el aprendizaje autodidacta y afianzar la confianza para encarar nuevos desafíos. Cualidad que estimo será de mucha utilidad en el futuro, cuando me inserte en el campo laboral.

Referencias

[1] E. Duman, M. Uysal, A.F. Alkaya, *"Migrating birds optimization: a new metaheuristic approach and its performance on quadratic assignment problem,"* Inform. Sci. 217 (2012) 65–77.

[2] Eduardo Lalla-Ruiz, Christopher Expósito-Izquierdo, Jesica de Armas, Belén Melián-Batista, and J. Marcos Moreno-Vega, *"Migrating Birds Optimization for the Seaside Problems at Maritime Container Terminals,"* Hindawi Publishing Corporation, Journal of Applied Mathematics, Volume 2015, Article ID 781907, 12 pages, http://dx.doi.org/10.1155/2015/781907.

[3] Vahit Tongur, Erkan Ülker, *"Migrating Birds Optimization for Flow Shop Sequencing Problems,"* Journal of Computer and Communications, 2014, 2, 142-147

[4] C.J. Cutts, J.R. Speakman, *"Energy savings in formation flight of pink-footed geese,"* Journal of Experimental Biology 189 (1994) 251–261.

[5] P.B.S. Lissaman, C.A. Shollenberger, *"Formation flight of birds,"* Science 168 (1970) 1003–1005.

[6] J.M.V. Rayner, *"A new approach to animal flight mechanics,"* Journal of Experimental Biology 80 (1979) 17–54.

[7] M. Andersson, J. Wallander, *"Kin selection and reciprocity in flight formation,"* Behavioral Ecology 15 (1) (2004) 158–162.

[8] F.R. Hainsworth, *"Precision and dynamics of positioning by Canada geese flying in formation,"* Journal of Experimental Biology 128 (1987) 445–462

[9] Quan-Ke Pan, Yan Dong, *"An improved migrating birds optimisation for a hybrid flowshop scheduling with total flowtime minimisation,"* Information Sciences 277 (2014) 643–655.

[10] Ekrem Duman, Mitat Uysal, Ali Fuat Alkaya *"Migrating Birds Optimization: A new metaheuristic approach and its performance on a quadratic assignment problem,"* Information Sciences 217 (2012) 65–77

[11] Vahit Tongur, Erkan Ülker, *"Migrating Birds Optimization for Flow Shop Sequencing Problems,"* Journal of Computer and Communications, 2014, 2, 142-147

[12] Ali Fuat Alkaya, Ramazan Algin, *"Metaheuristic based solution approaches for the obstacle neutralization problem,"* Expert Systems with Applications 42 (2015) 1094–1105.

[13]] Sadegh Niroomand, Abdollah Hadi-Vencheh, Ramazan Şahin, Béla *Vizvári "Modified migrating birds optimization algorithm for closed loop layout,"* Expert Systems with Applications 42 (2015) 6586–6597.

[14] Eduardo Lalla-Ruiz, Christopher Expósito-Izquierdo, Jesica de Armas, Belén Melián-Batista, and J. Marcos Moreno-Vega, *"Migrating Birds Optimization for the Seaside Problems at Maritime Container Terminals,"* Hindawi Publishing Corporation, Journal of Applied Mathematics, Volume 2015, Article ID 781907, 12 pages, http://dx.doi.org/10.1155/2015/781907.

[15] E. Lalla-Ruiz, B. Meli´ an-Batista, and J. Marcos Moreno-Vega, *"Artificial intelligence hybrid heuristic based on tabu search for the dynamic berth allocation problem,"* Engineering Applications of Artificial Intelligence, vol. 25, no. 6, pp. 1132–1141, 2012.

[16] G. Dantzig and J. Ramser, *"The Truck Dispatching Problem,"* Management Science, Vol. 6, No. 1, 1959, pp. 80-91. doi:10.1287/mnsc.6.1.80.

[17] Toth, P., Vigo, D. (1998). *"Exact algorithms for vehicle routing."* In: Crainic, T., Laporte, G. (Eds.), Fleet Management and Logistics. Kluwer Academic, Boston, pp. 1–31.

[18] Padberg, M. and Rinaldi, *G. (1991). "A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems".* Siam Review: 60–100. Doi: 10.1137/1033004.

[19] Gulay Barbarosoglu, Demet Ozgur, *"A tabu search algorithm for the vehicle routing problem,"* Computers & Operations Research 26 (1999) 255-270

[20] E. Aziz, *"An Algorithm for the Vehicle Problem,"* Interna- tional Journal of Advanced Robotic Systems, Vol. 7, No. 2, 2010, pp. 125-132.

[21] R. Montemanni, L. M. Gambardella, A. E. Rizzoli and A. V. Donati, *"Ant Colony System for a Dynamic Vehicle Routing Problem,"* Journal of Combinatorial Optimization, Vol. 10, No. 4, 2005, pp. 327-343. Doi:10.1007/s10878-005-4922-6

[22] Jinhui Yang, Chunguo Wu, Heow Pueh Lee, Yanchun Liang, *"Solving traveling salesman problems using generalized chromosome genetic algorithm,"* Elsevier, Progress in Natural Science 18 (2008) 887–892.

[23] Gilbert Laporte, *"The Traveling Salesman Problem: An overview of exact and approximate algorithms,"* Centre de recherche sur les transports, European Journal of Operational Research 59 (1992)231-247.

[24] M. Fisher and R. Jaikumar, *"A Generalized Assignment Heuristic for Vehicle Routing,"* Networks, Vol. 11, No. 2, 1981, pp. 109-124. doi:10.1002/net.3230110205

[25] M. Gandreau, F. Guertin, J.-Y. Potvin and R. Seguin, *"Neighborhood Search Heuristics for a Dynamic Vehicle Dispatching Problem with Pick-Ups and Deliveries,"* Transportation Research Part E: Logistics and Transportation Review, Vol. 14, No. 3, 2006, pp. 157-174.

[26] Suresh Nanda Kumar, Ramasamy Panneerselvam, *"A Survey on the Vehicle Routing Problem and Its Variants,"* Intelligent Information Management, 2012, 4, 66-74.

[27] Emmanouil E. Zachariadis, Christos D. Tarantilis, Christos T. Kiranoudis, *"A Guided Tabu Search for the Vehicle Routing Problem with two-dimensional loading constraints,"* European Journal of Operational Research 195 (2009) 729–743.

[28] J. L. Blanton and R. L. Wainwright, *"Multiple Vehicle Routing with Time and Capacity Constraints Using Genetic Algorithms,"* Proceedings of the 5th International Intelligence 1418, Conference on Genetic Algorithms, San Francisco, 17 July 1993, pp. 452-459.

[29] J. Berger, M. Salois and R.A. Begin, *"Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows,"* Lecture Notes in Artificial Intelligence 1418,Springer, Berlin, 1998, pp. 114-127.

[30] J.-Y. Potvin and S. Bengio, *"The Vehicle Routing Problem with Time Windows Part II: Genetic Search,"* INFORMS Journal on Computing, Vol. 8, No. 2, 1996, pp. 165-172. doi:10.1287/ijoc.8.2.165.

[31] S. R. Thangiah, I. H. Osman and T. Sun, *"Hybrid Genetic Algorithms, Simulated Annealing and Tabu Search Meth- ods for Vehicle Routing Problems with Time Windows,"* Technical Report UKC/OR94/4, Institute of Mathematics and Statistics, University of Kent,

[32] W.-K. Ho, J. C. Agn and A. Lim, *"A Hybrid Search Algorithm for the Vehicle Routing Problem with Time Windows,"* International Journal on Artificial Intelligence Tools, Vol.10, No. 3, 2001, pp. 431-449. Doi: 10.1142/S021821300100060X

[33] J. Berger, M. Barkaoui and O.A. Bräysy, *"Route-Directed Hybrid Genetic Approach for the Vehicle Routing Problem with Time Windows,"* INFOR, Vol. 41, No. 2, 2003, pp. 179-194.

[34] C. L. Hoong, M. Sim and K. M. Teo, *"Vehicle Routing Problem with Time-Windows and a Limited Number of Vehicles,"* European Journal of Operational Research, Vol. 148, No. 3, 2003, pp. 559-569., doi:10.1016/S0377-2217(02)00363-6.

[35] Yu Bin and Yang Zhong Zhen,*"An Ant Colony Optimization Model: The Period Vehicle Routing Problem with Time Windows,"* Transportation Research Part E, Vol. 47, 2011, pp. 166-181. doi:10.1016/j.tre.2010.09.010.

[36] E. Alba and B. Dorronsoro, *"The Exploration/Exploitation Tradeoff in Dynamic Cellular Genetic Algorithms"*, IEEE Transactions on Evolutionary Computation, Vol. 9, No. 2, 2005, pp. 126-142. doi:10.1109/TEVC.2005.843751.

[37] C. Benoit., J.-F. Cordeau and G. Laporte, *"The Multi-Depot Vehicle Routing Problem with Inter-Depot Routes"*, European Journal of Operational Research, Vol. 176, No. 2, 2007, pp. 756-773. doi:10.1016/j.ejor.2005.08.015.

[38] Yves Rochat, Éric D. Taillard, *"Probabilistic Diversification and Intensification in Local Search forVehicle Routing", CRT–95–13, ORWP 95/03, October 1995. To appear in Journal of heuristics 1, 1995, pp. 147 – 167.*

[39] Eduardo Uchoa, Diego Pecin, Artur Pessoa, Marcus Poggi, Anand Subramanian, Thibaut Vidal, *"New Benchmark Instances for the Capacitated Vehicle Routing Problem"*