ICT-UNPA-58-2013
ISSN: 1852-4516
Aprobado por Resolución N° 0579/13-R-UNPA
http://dx.doi.org/10.22305/ict-unpa.v5i1.67

Algoritmos Multirecombinativos aplicados al Problema de Ruteo de Vehículos

Renzo Miño[1] y Mg. Andrea Villagra[2]
[1]{Becario de Investigación}
[2]{Docente Investigador de la UNPA}
{rminio, avillagra}@uaco.unpa.edu.ar

UNPA – UACO
Universidad Nacional de la Patagonia Austral – Unidad Académica Caleta Olivia
Departamento de Ciencias Exactas y Naturales
LabTEm- Laboratorio de Tecnologías Emergentes
Caleta Olivia, 2012

RESUMEN

El diseño de algoritmos eficientes para resolver problemas complejos ha sido tradicionalmente uno de los aspectos más importantes en la investigación en el campo de la informática. El objetivo perseguido en este campo es fundamentalmente el desarrollo de nuevos métodos capaces de resolver problemas complejos con el menor esfuerzo computacional posible, mejorando así a los algoritmos existentes. En consecuencia, esto no sólo permite afrontar los problemas de forma más eficiente, sino afrontar tareas vedadas en el pasado debido a su alto costo computacional. Las metaheurísticas son métodos que integran procedimientos de mejora local y estrategias de alto nivel para realizar una búsqueda robusta en el espacio-problema. El problema de ruteo de vehículos es un problema de optimización combinatoria de gran importancia en diferentes entornos logísticos debido a su dificultad (NP-duros) y a sus múltiples aplicaciones industriales. Se han propuestos varias soluciones a este problema haciendo uso de heurísticas y metaheurísticas. En este trabajo proponemos dos algoritmos para resolver el problema de ruteo de vehículos con capacidad limitada, utilizando como base un Algoritmo Evolutivo conocido como MCMP-SRI *(Stud and Random Inmigrants*) combinados con *Hill- Climbing*. Detalles de los algoritmos y los resultados de los experimentos muestran un promisorio comportamiento para resolver el problema.

Palabras Claves: Problema de ruteo de vehículos, Algoritmos Evolutivos multirecombinativos.

1. INTRODUCCIÓN

La optimización es una disciplina fundamental en campos de las ciencias tales como Informática, Inteligencia Artificial, Logística, Biología, Tecnología de la Producción, Física, etc. El concepto de optimización puede verse como el proceso de encontrar y mejorar el rendimiento de una aplicación o dispositivo a partir de determinados cambios lógicos o físicos. Un problema de optimización se formaliza como un par (S, f), donde S < > 0 representa el espacio de soluciones (o de búsqueda) del problema, mientras que f es un criterio de calidad conocido como función objetivo, definida como:

$$f: S \rightarrow R.$$

Así, resolver un problema de optimización consiste en encontrar un conjunto de valores adecuados de forma que la solución representada por estos valores $i^* \in S$ satisfaga la siguiente desigualdad:

$$f(i^*) \leq f(i), \ \forall \ i \in S$$

Asumir el caso de maximización o minimización no restringe en ningún caso la generalidad de los resultados, puesto que se puede establecer una igualdad entre tipos de problemas de maximización y minimización [Bäck 1996], [Goldberg 1989].
A este tipo de problemas se los puede dividir en dos categorías [Blum y Roli 2003], [Papadimitriou y Steiglitz 1998]: aquellos en los que la solución está codificada mediante valores reales y aquellos cuya solución está codificada con valores enteros.
En los últimos encontramos los problemas denominados de optimización combinatoria. Algunos ejemplos muy conocidos son el problema del viajante de comercio (TSP - Travelling Salesman Problem), el problema de asignación cuadrática (QAP - Quadratic Assignment Problems) o los problemas de planificación (Scheduling Problems), entre otros, [Cook et al. 1998], [Papadimitriou y Steiglitz 1998].
En la literatura existen multitud de propuestas de técnicas algorítmicas, tanto exactas como aproximadas para resolver problemas de optimización. Los algoritmos exactos garantizan encontrar una solución óptima al problema para toda instancia de tamaño finito [Hochbaum 1996], [Korf 1985], [Russell y Norvig 1995], [Vazirani 2003]. Sin embargo, los métodos exactos necesitan tiempos exponenciales de computación cuando se trata con instancias grandes de problemas complejos. Los problemas NP − completos no tienen un algoritmo en tiempo polinómico que los resuelva. También existe otro tipo de problemas al menos tan difíciles de resolver como los anteriores denominados NP − duros para los cuales tampoco existe un algoritmo polinómico que los resuelva, es decir que esta clase de problemas NP no puede abordarse de forma realista con técnicas exactas. En consecuencia, el uso de técnicas aproximadas está recibiendo en las últimas décadas cada vez más atención. En estos métodos aproximados se sacrifica la garantía de encontrar el óptimo global al problema (en muchos casos, aunque no siempre) con el fin de encontrar soluciones buenas en un tiempo significativamente reducido en comparación con los métodos exactos.
En las dos últimas décadas ha emergido un nuevo tipo de técnicas aproximadas que consiste básicamente en la combinación de métodos heurísticos (técnicas aproximadas con componentes aleatorios guiados) básicos en entornos de más alto nivel con el fin de explorar el espacio de búsqueda de una forma eficiente y efectiva. Estos métodos son comúnmente conocidos con el término metaheurísticas. En [Blum y Roli 2003] se pueden encontrar recopiladas varias definiciones de metaheurísticas dadas por diferentes autores, pero en

general podemos decir que las metaheurísticas son estrategias de alto nivel que planifican de manera estructurada la aplicación de varias operaciones para explorar espacios de búsqueda de elevada dimensión y complejidad intrínseca.

El problema de ruteo de vehículos (VRP su sigla en inglés) es un problema de optimización combinatoria de gran importancia en el campo del transporte, distribución y logística [Dantzig y Ramser 1959]. El VRP es un nombre genérico dado a una gran clase de problemas, donde un conjunto de rutas para una flota de vehículos basados en uno o varios depósitos, debe ser determinado para un número de clientes o ciudades geográficamente dispersadas. El problema consiste en asignar a cada vehículo una ruta de clientes, de manera que se minimice el costo de transporte. En la Figura 1 se muestra un ejemplo clásico de VRP.

Este problema, VRP, ha sido catalogado como NP-Completo [Lenstra y Kan 1981] debido a la gran cantidad de consumo en recursos computacionales para encontrar una solución óptima que crece de forma exponencial con respecto al tamaño del problema. Este tipo de problema es tratado con algoritmos que no tienen la necesidad de explorar todo el espacio de búsqueda asociado para dar una respuesta aproximada. Los algoritmos metaheurísticos son una familia de algoritmos cuya meta es precisamente dar soluciones aproximadas a problemas generales de tipo NP, sin necesidad de recorrer todo el espacio de búsqueda.

Existen diferentes variantes del problema de ruteo de vehículos, que incluyen restricciones adicionales y la incorporación de múltiples variables.

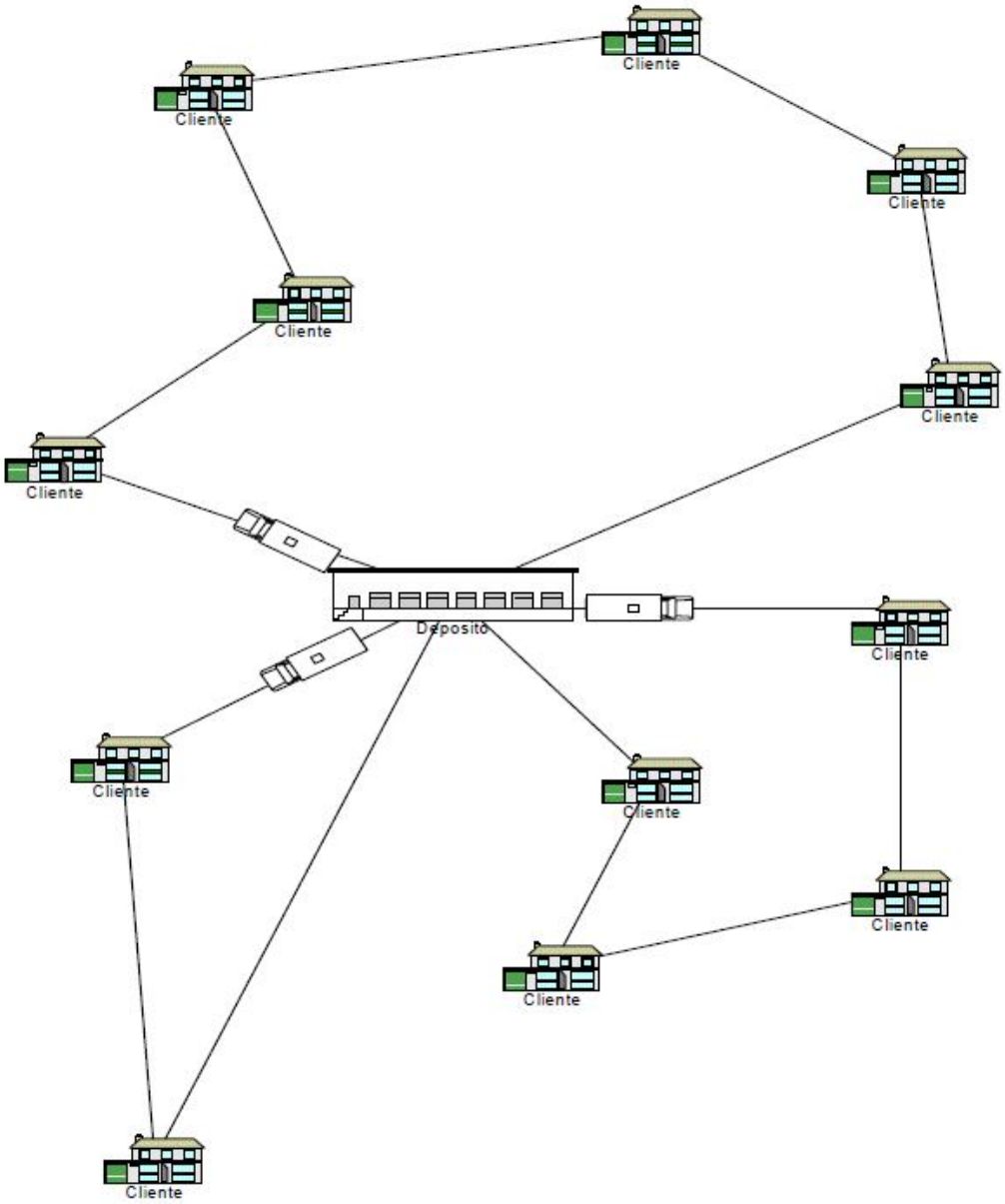

Figura 1: Ejemplo de VRP Clásico.

El objetivo de este trabajo es proponer algoritmos multirecombinativos eficientes aplicados a resolver el problema de ruteo de vehículos.

2. MARCO TEÓRICO

El VRP es un problema de optimización combinatoria, consiste en servir una serie de clientes ubicados geográficamente de manera dispersa, para atender los clientes se cuenta con una flota de vehículos que parten desde un depósito central, el problema consiste en asignar a cada vehículo una ruta de clientes, de manera que se minimice el costo de transporte.

Cada cliente tiene cierta demanda que deberá ser satisfecha por algún vehículo. En muchos casos, la demanda es un bien que ocupa lugar en los vehículos y es usual que un mismo vehículo no pueda satisfacer la demanda de todos los clientes en una misma ruta. Un caso equivalente al anterior ocurre cuando los clientes son proveedores y lo que se desea es recoger la mercadería y transportarla hacia el depósito. También podría ocurrir que la mercadería deba ser transportada a los clientes pero no esté inicialmente en el depósito, sino distribuida en ciertos sitios proveedores. En este caso, los proveedores deben ser visitados antes que los clientes.

En otros casos la demanda no es un bien sino un servicio: el cliente simplemente debe ser visitado por el vehículo. Un mismo vehículo podría, potencialmente, visitar a todos los clientes. En otra variante del problema, cada cliente tiene una ubicación y desea ser transportado hacia otro sitio. Aquí la capacidad del vehículo impone una cota sobre la cantidad de clientes que puede alojar simultáneamente.

Es usual que cada cliente deba ser visitado exactamente una vez. Sin embargo, en ciertos casos se acepta que la demanda de un cliente sea satisfecha en momentos diferentes y por vehículos diferentes.

Los clientes podrían tener restricciones relativas a su horario de servicio. Usualmente estas restricciones se expresan en forma de intervalos de tiempo (llamados ventanas de tiempo) en los que se puede arribar al cliente.

Tanto los vehículos como las mercaderías a distribuir (si las hubiera) suelen estar ubicadas en depósitos. Usualmente se exige que cada ruta comience y finalice en un mismo depósito, aunque este podría no ser el caso en algunas aplicaciones (por ejemplo, podría ser que el viaje debiera finalizar en el domicilio del conductor del vehículo).

En los problemas con múltiples depósitos cada uno de estos tiene diferentes características, por ejemplo, su ubicación y capacidad máxima de producción. Podría ocurrir que cada depósito tenga una flota de vehículos asignada a priori o que dicha asignación sea parte de lo que se desea determinar.

Los depósitos, al igual que los clientes, podrían tener ventanas de tiempo asociadas. En algunos casos debe considerarse el tiempo necesario para cargar o preparar un vehículo antes de que comience su ruta, o el tiempo invertido en su limpieza al regresar. Incluso, por limitaciones de los propios depósitos, podría querer evitarse que demasiados vehículos estén operando en un mismo depósito a la vez (es decir, la congestión del depósito).

La capacidad de un vehículo podría tener varias dimensiones, como por ejemplo peso y volumen. Cuando en un mismo problema existen diferentes mercaderías, los vehículos podrían tener compartimentos, de modo que la capacidad del vehículo dependa de la mercadería de que se trate. En general, cada vehículo tiene asociado un costo fijo en el que se incurre al utilizarlo y un costo variable proporcional a la distancia que recorra.

Los problemas en que los atributos (capacidad, costo, etc.) son los mismos para todos los vehículos se denominan de flota homogénea, y, si hay diferencias, de flota heterogénea. La cantidad de vehículos disponibles podría ser un dato de entrada o una variable de decisión. El objetivo más usual suele ser utilizar la menor cantidad de vehículos y minimizar la distancia recorrida ocupa un segundo lugar.

Regulaciones legales podrían imponer restricciones sobre el tiempo máximo que un vehículo puede estar en circulación e incluso prohibir el pasaje de ciertos vehículos por ciertas zonas. En algunos casos se desea que la cantidad de trabajo realizado por los vehículos (usualmente el tiempo de viaje) no sea muy dispar.

Seguidamente se detallan algunas de las variantes más conocidas del VRP:

- Problema con Capacidades (*Capacitated* VRP-CVRP). El problema que analizamos usa este tipo de VRP.

- Problema con Ventanas de Tiempo (VRP *with Time Windows* - VRPTW)
- Problema con Múltiples Depósitos (*Multiple Depot* VRP-MDVRP)
- Problema con Entregas y Devoluciones (VRP *with Pickup and Delivery* -
- VRPPD)
- Problema de entregas divididas con diferentes vehículos (*Split Delivery*
- VRP-SDVRP)
- Problema con diferentes valores al azar como el número de clientes, tiempo de servicio, tiempo de recorrido, etc. (*Stochastic* VRP-SVRP)

Las metaheurísticas han sido aplicadas para distintas variantes del problema de VRP, por este motivo a continuación se realiza una introducción a las mismas.

Las metaheurísticas son estrategias inteligentes para diseñar o mejorar procedimientos heurísticos muy generales con un alto rendimiento, además de proveer métodos para explorar el espacio de búsqueda.

Hay diferentes formas de clasificar y describir las técnicas metaheurísticas [Crainic y Toulouse 2003]. Dependiendo de las características que se seleccionen se pueden obtener diferentes taxonomías: basadas en la naturaleza o no basadas en la naturaleza, basadas en memoria o sin memoria, con función objetivo estática o dinámica, etc. En este trabajo se ha elegido clasificarlas de acuerdo a si en cada paso manipulan un único punto del espacio de búsqueda o trabajan sobre un conjunto (población) de ellos, es decir, esta clasificación divide a las metaheurísticas en basadas en trayectoria y basadas en población. Esta clasificación es ampliamente utilizada en la comunidad científica y se la muestra de forma gráfica en la Figura 2. Las siglas se corresponden de la siguiente manera: *Simulated Annealing (SA)* [Kirkpatrick et al. 1983], *Tabu Search (TS))* [Glover y Laguna 1993], *Greedy Randomized Adaptive Search Procedure (GRASP)* [Feo y Resende 1995], *Variable Neighborhood Search (VNS)* [Hansen y Mladenovic 1997], *Iterated Local Search (ILS)* [Louren et al. 2001], *Evolutionary Algorithms (EA)* [Bäck et al. 1997], *Scatter Search (SS)* [Glover y Kochenberger 2002], *Ant Colony Optimization (ACO)* [Dorigo 1992; Dorigo y Di Caro 1999], *Particle Swarm Optimization (PSO)* [Eberhart y Kennedy 1995].

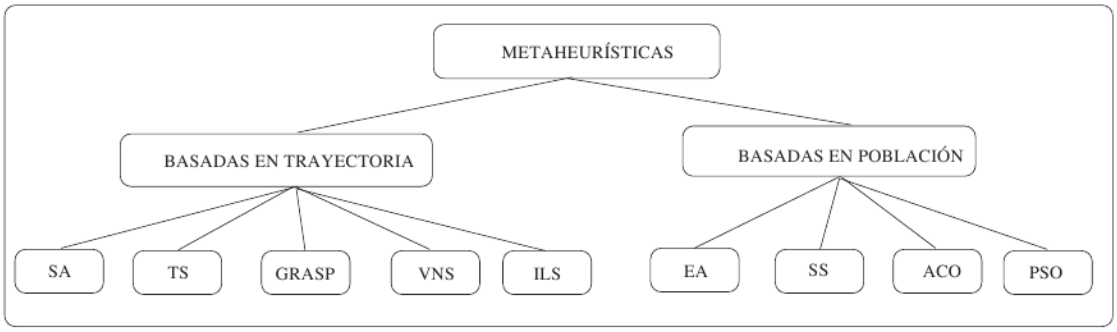

Figura 2: Clasificación de las Metaheurísticas.

3. ALGORITMOS UTILIZADOS PARA EL PROBLEMA DE CVRP

Para resolver el problema de ruteo de vehículos con capacidad limitada se proponen dos versiones basadas en MCMP-SRI aplicando Hill-Climbing.

Este algoritmo MCMP-SRI (Multiple Crossover Multiples Parents – Stud and Random Inmigrates) fue aplicado en diferentes problemas de planificación de máquina única para

casos estáticos y casos dinámicos y los resultados obtenidos fueron satisfactorios. En los problemas estáticos, MCMP-SRI se aplicó para resolver problemas de Earliness y Tardiness [Pandolfi et al. 2001], Weighted Tardiness [De San Pedro et al. 2001], Average Tardiness [Pandolfi et al. 2003] y Weighted Number of Tardy Jobs [De San Pedro et al. 2003]. En los problemas dinámicos MCMP-SRI fue aplicado para resolver problemas de adaptabilidad para Earliness y Tardiness [Villagra et al. 2001], en problemas de dinámica parcial y total para Weighted Tardiness [Lasso et al. 2004] y Average Tardiness [De San Pedro et al. 2003]. Además fue utilizado en el manejo de restricciones con metaheurísticas en [Villagra et al. 2012] con resultados promisorios.

El funcionamiento de MCMP-SRI para crear descendientes, en es el siguiente: de la vieja población de individuos, se selecciona un individuo, el semental (*Stud*), a través de selección proporcional. Se genera un pool de apareamiento con n2 padres generados aleatoriamente. El semental se aparea con cada padre del pool de apareamiento y las parejas se someten a operaciones de recombinación, y generan 2*n2 descendientes. El mejor de los 2*n2 descendientes, se almacena en un pool de hijos temporal. Esta operación de recombinación se repite n1 veces, para diferentes puntos de corte cada vez, hasta que el pool de hijos se complete. Finalmente, el mejor descendiente creado de n2 padres y n1 operaciones de recombinación, se inserta en la nueva población. El método de recombinación utilizado fue PMX (Partial Mapped Crossover): [Goldberg y Lingle 1987] que puede verse como una extensión del cruzamiento de dos puntos para representaciones basadas en permutaciones. La selección de individuos fue a través de selección proporcional. En la Figura 3 se muestra el código del algoritmo MCMP-SRI.

```
MCMP-SRI
t=0; {Generación Inicial}
inicializar (Stud(t));
evaluar (Stud(t));
while (not max_evaluaciones) do
        pool_apareamiento = Inmigrates_generados_aleatoriamente ∪ Select (Stud(t));
        while (not max_padres) do
                while (not max_recombinaciones) do
                        evalucionar (pool_apareamiento);
                end while
        end while
        evaluar (mating_pool);
        Stud(t+1) = seleccionar la nueva población del pool de apareamiento.
        t = t+1;
end while
```

Figura 3: Algoritmo MCMP-SRI

En cuanto al algoritmo de Hill Climbing (HC) [Hoos y Stützle 2004] puede considerarse como una metaheurística de trayectoria ya que es una versión básica de un algoritmo de búsqueda local en donde se considera todo el vecindario. Se trata de un algoritmo de descenso. Es decir, que la elección de las dirección de búsqueda se hace de forma exhaustiva (se calculan todas las posibles direcciones), y se elige aquella que consigue un mayor descenso.

Este algoritmo usa una técnica de mejoramiento iterativo. Comienza a partir de un punto (punto actual) en el espacio de búsqueda. Si el nuevo punto es mejor, se transforma en el punto actual, si no, otro punto vecino es seleccionado y evaluado. El algoritmo se debe completar con un criterio de parada. En la Figura 4 se muestra el algoritmo de Hill-Climbing.

```
HILL CLIMBING

S← Solución inicial
repeat
    R ← Vecindario(S)  {Genera un vecindario y devuelve la mejor}
    if F(R) > F(S) then
        S ← R
until condicion_de_parada
return S
```

Figura 4: Algoritmo de Hill-Climbing

Algoritmos propuestos

A continuación se presentan los dos algoritmos propuestos. Estas dos versiones han sido seleccionadas luego realizar diferentes propuestas variando la aplicación de HC, es decir aplicar HC a toda la solución o aplicar HC a cada ruta y además variar en cada caso la cantidad de vecinos utilizados por HC.

En la primera versión denominada MCMP-SRI-HC se aplica Hill-Climbing a cada ruta en cada generación. Es decir, que luego de finalizada una generación del MCMP-SRI se toma la mejor solución se le aplica HC a cada ruta y si el HC encuentra una mejor solución, ésta pasa a reemplazar la anterior; caso contrario se mantiene la mejor solución de esa generación. Este proceso se repite para cada generación. La Figura 5 muestra la versión propuesta MCMP-SRI-HC.

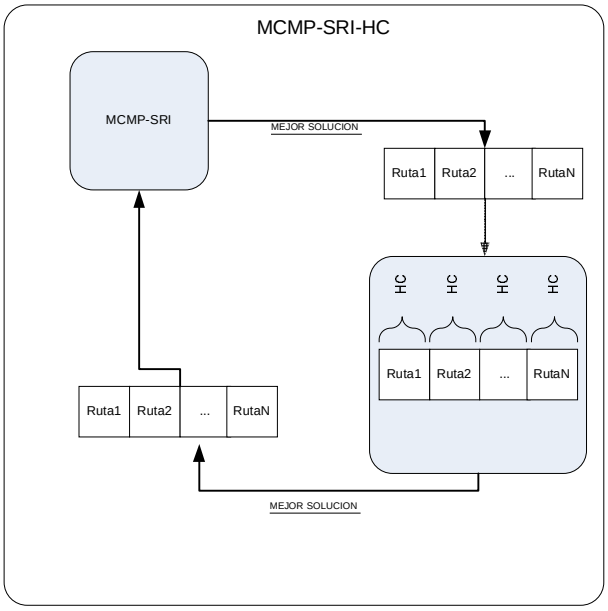

Figura 5: Mecanismo de MCMP-SRI-HC

En la segunda versión propuesta denominada MCMP-SRI-Comb se incorpora auto-adaptación al algoritmo ya que toma una decisión y aplica HC a cada una de las rutas que componen la solución (como en la versión anterior) o HC a toda la solución. El proceso es el siguiente: luego de finalizada una generación del MCMP-SRI se toma la mejor solución y se

la compara con la mejor solución global, si esta solución es mejor que la mejor solución global entonces se le aplica HC a cada ruta y si el HC encuentra una mejor solución, ésta pasa a reemplazar la anterior; caso contrario se mantiene la mejor solución de esa generación. En caso de que esta solución no sea mejor que la mejor solución global entonces se le aplica HC global es decir, se le aplica HC a la solución como un todo. De igual forma que antes si el HC encuentra una mejor solución, ésta pasa a reemplazar la anterior; caso contrario se mantiene la mejor solución de esa generación. Este proceso se repite para cada generación. La Figura 6 muestra esta versión MCMP-SRI-Comb.

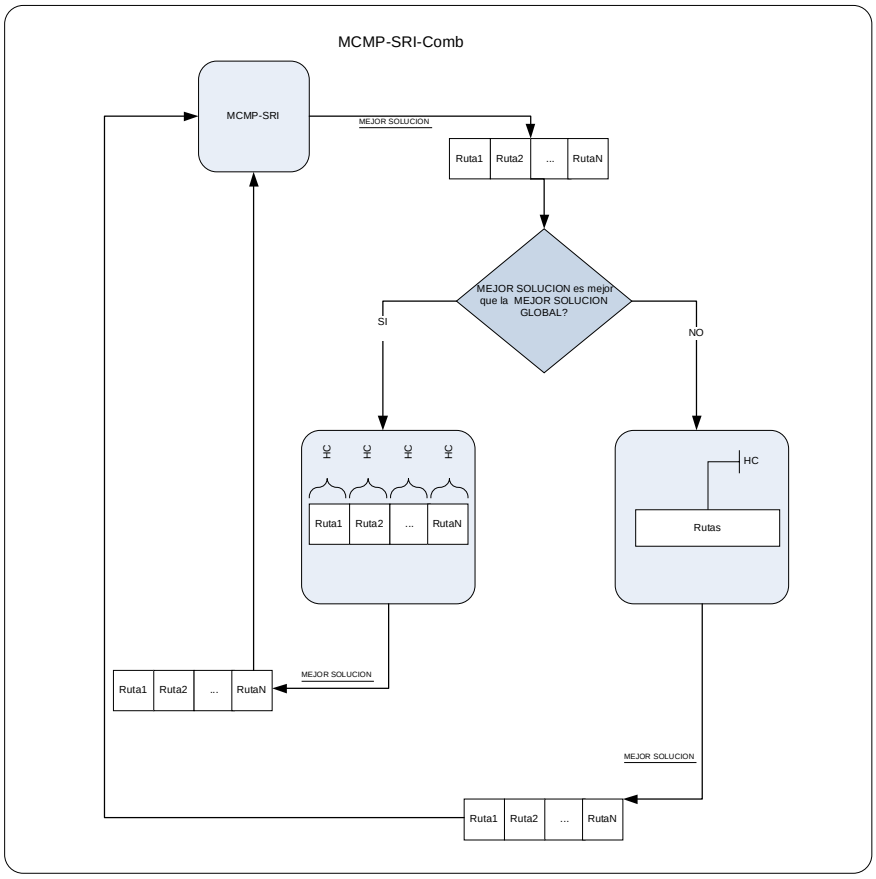

Figura 6: Mecanismo de MCMP-SRI-Comb

4. DISEÑO DE EXPERIMENTOS Y RESULTADOS

La metodología utilizada para el desarrollo de este trabajo se realizó siguiendo las fases típicas que establece la filosofía del método científico [Gauch 2002], [Whewell y Butts 1989] según definió F. Bacon: (1) Observación: Observar es aplicar atentamente los sentidos a un objeto o a un fenómeno, para estudiarlos tal como se presentan en realidad. (2) Inducción: La acción y efecto de extraer, a partir de determinadas observaciones o experiencias, el principio particular de cada una de ellas. (3) Hipótesis: Planteamiento mediante la observación siguiendo las normas establecidas por el método científico. (4) Probar la hipótesis por experimentación. (5) Demostración o refutación de la hipótesis: Analizar los resultados obtenidos con los nuevos modelos desarrollados, comparándolos exhaustivamente con los

resultados de los algoritmos canónicos existentes. (6) Conclusiones: Presentar las conclusiones arribadas.

Para analizar la performance de los algoritmos se utilizaron las instancias provistas por Augerat et al[1]. Se realizaron 30 corridas independientes de todos los algoritmos de las cuales se seleccionaron las 10 instancias más representativas y se compararon los resultados de nuestras versiones con los resultados obtenidos por un Algoritmo Genético Simple (AG) y MCMP-SRI.

A continuación se describen las instancias para el problema de Augerat et al. Esta batería de problemas, propuesta en 1995, se compone de tres conjuntos de problemas (conjuntos A, B y P). Todas las instancias de cada conjunto tienen distintas características, como la distribución de la posición de los clientes. Se ha comprobado que las mejores soluciones conocidas son los óptimos para cada instancia de este conjunto de problemas.

A continuación se describe el **Conjunto A** utilizado en este trabajo: Este conjunto está constituido por instancias donde tanto las posiciones de los clientes como las demandas se generan aleatoriamente mediante una distribución uniforme. El tamaño de las instancias está en el rango de 31 a 79 clientes.

Parametrización y Resultados

En esta sección se presentan los detalles y resultados de los métodos aplicados en este trabajo para resolver el CVRP, con el objetivo de encontrar una distancia mínima total de viaje de un plan de rutas que satisface la restricción de capacidad del vehículo. En cada uno de los algoritmos, para codificar las visitas a los clientes que representan una posible solución, se utilizó una permutación de números enteros. Donde cada permutación $p = (p1, p2, \dots, pn)$ es un cromosoma en el cual pi representa el cliente i que debe ser visitado y n representa la cantidad de clientes a visitar. El cromosoma establece el orden de la secuencia a seguir para visitar cada cliente.

La función objetivo es minimizar la distancia total del plan de rutas general de manera de satisfacer las demandas de todos los clientes, cumpliendo con la restricción de capacidad del vehículo, Q.

Tamaño población	15
Tamaño cromosoma (cantidad de clientes)	75
Criterio parada (generación)	5000
Recombinación	PMX
Mutación	SW
Probabilidad Recombinación	0,65
Probabilidad Mutación	0,05
Nº de recombinación (n1)(*)	16
Nº de padres (n2)(*)	18
Nº de vecinos (**)	20

Tabla 1: Parámetros de los Algoritmos Utilizados

(*) El número recombinación y de padres utilizamos en los algoritmos multirecombinativos

(**) El número de vecinos sólo se aplica cuando utilizamos los algoritmos propuestos

[1] http://www.coin-or.org/SYMPHONY/branchandcut/VRP/data/index.htm.old

Se utilizó un tamaño de población de 15 individuos. La población inicial se generó aleatoriamente. Se estableció la probabilidad de mutación en 0,05 y la probabilidad de recombinación en 0,65. El número n1 (número de operaciones de recombinación) y n2 (número de padres) se estableció en 16 y 18 respectivamente, estos parámetros seleccionaron en base a la experimentación de los valores previamente usados exitosamente [Lasso et al. 2004]. En la Tabla 1 se presenta los parámetros utilizados.
En la Tabla 2 se muestra el resumen de las corridas de las 10 instancias más representativas aplicándole los algoritmos mencionados anteriormente. Las dos primeras columnas corresponden a la instancia utilizada y a su valor optimo. Luego para cada uno de los algoritmos se muestra la mediana y el error porcentual con respecto al valor optimo. Podemos observar una clara diferencia en los resultados obtenidos entre el algoritmo AG y las tres versiones multirecombinativas.

		AG		MCMP-SRI		MCMP-SRI-HC		MCMP-SRI- Comb	
Instancia	Óptimo	Mediana	Error	Mediana	Error	Mediana	Error	Mediana	Error
A-n33-k5	661,00	863,72	0,31	706,20	0,07	705,21	0,07	**692,99**	**0,05**
A-n33-k6	742,00	895,87	0,21	**770,61**	**0,04**	778,72	0,05	781,14	0,05
A-n34-k5	778,00	970,02	0,24	833,41	0,07	836,64	0,07	841,48	0,08
A-n37-k6	949,00	1139,65	0,21	1004,87	0,07	1019,86	0,08	1016,67	0,07
A-n45-k7	1146,00	1396,02	0,22	1235,81	0,08	1253,75	0,09	1252,50	0,09
A-n53-k7	1010,00	1467,17	0,46	1177,63	0,16	1189,02	0,18	1180,04	0,17
A-n61-k9	1035,00	1449,55	0,41	1214,39	0,16	1208,56	0,17	1194,54	0,15
A-n62-k8	1290,00	1840,78	0,42	1487,72	0,16	1511,99	0,18	1532,15	0,19
A-n64-k9	1402,00	1881,99	0,35	1600,69	0,14	1625,44	0,16	1601,42	0,14
A-n80-k10	1764,00	2553,64	0,45	2072,06	0,18	2116,96	0,21	2085,30	0,18

Tabla 2: Resultados obtenidos por los algoritmos para 10 instancias de problemas

	AG	MCMP-SRI	MCMP-SRI-HC	MCMP-SRI-Comb
Instancia	Error	Error	Error	Error
A-n33-k5	0,31	0,07	0,07	**0,05**
A-n33-k6	0,21	**0,04**	0,05	0,05
A-n34-k5	0,24	**0,07**	**0,07**	0,08
A-n37-k6	0,21	**0,07**	0,08	**0,07**
A-n45-k7	0,22	**0,08**	0,09	0,09
A-n53-k7	0,46	**0,16**	0,18	0,17
A-n61-k9	0,41	**0,16**	0,17	0,15
A-n62-k8	0,42	**0,16**	0,18	0,19
A-n64-k9	0,35	**0,14**	0,16	**0,14**
A-n80-k10	0,45	**0,18**	0,21	**0,18**

Tabla 3: Porcentaje de Error obtenidos por los algoritmos AG, MCMP-SRI, MCMP-SRI-HC y MCMP-SRI-Comb para 10 instancias de problemas

Para realizar un análisis más profundo de los resultados únicamente se analiza el error porcentual. En la Tabla 3 se muestran los resultados utilizados para realizar los análisis estadísticos. Podemos observar que en 9 de 10 instancias el menor porcentaje de error fue obtenido por MCMP-SRI, mientras que el algoritmo MCMP-SRI-Comb mejoró a MCMP-SRI en la primer instancia y en cuatro instancias igualó el porcentaje de error obtenido por MCMP-SRI. No obstante, para poder concluir sobre si existen diferencias estadísticamente significativas entre alguno de los algoritmos debemos realizar otros test estadísticos.

Como primer paso debemos determinar que tipo de test se debe aplicar, y por lo tanto se debe verificar Independencia, Normalidad y Homocedasticidad para aplicar test paramétrico.
La Independencia de los resultados es obvia, ya que estamos trabajando con los resultados de corridas independientes.
En las Tablas 4, 5 y 6 mostramos los resultados obtenidos para los test de Normalidad y homocedasticidad. A continuación mostramos los test de Kolmogarov-Smirnov y Shapiro-Wilk, junto con un análisis de homocedasticidad utilizando el test de Levene. Para todos los test utilizamos una probabilidad de error p = 0, 05 y como herramienta SPSS. La Tabla 4 muestra los resultados obtenidos donde el símbolo "*" indica que no cumple la normalidad y el valor a continuación representa el valor p en cada caso. Analizando los resultados de la Tabla 4 sobre la aplicación del test de Kolmogorov- Smirnov podemos decir que cuando los resultados de al menos un algoritmo no cumple las condiciones de normalidad debemos entonces aplicar test no paramétricos. Es por eso que por ejemplo en la instancia A-n33-k5 los resultados del algoritmo MCMP-SRI-Comb no son normales por lo tanto se deberán aplicar test no paramétricos. Analicemos ahora el caso de la instancia de problema A-n34-k5 en este caso ambos test de normalidad se cumplen pero al aplicar el test de Levene (Tabla 6) observamos que los resultados obtenidos no cumplen con la propiedad de homocedasticidad, por lo tanto también se deben aplicar test no paramétricos en este caso. Finalmente, observando cada uno de los resultados obtenidos se puede determinar que para todos los casos se utilizarán test no paramétricos.

	AG	MCMP-SRI	MCMP-SRI-HC	MCMP-SRI-Comb
A-n33-k5	0,20	0,20	0,07	*0,03
A-n33-k6	0,20	*0,03	0,20	0,20
A-n34-k5	0,20	0,20	0,20	0,20
A-n37-k6	0,20	0,06	0,20	0,20
A-n45-k7	0,13	0,20	0,16	0,20
A-n53-k7	0,20	0,20	0,20	*0,00
A-n61-k9	0,20	0,20	0,20	0,20
A-n62-k8	0,80	0,14	0,11	0,07
A-n64-k9	0,20	0,20	0,20	0,20
A-n80-k10	0,20	0,20	*0,03	0,11

Tabla 4: Test de normalidad de Kolmogorov-Smirnov

La Tabla 5 muestra los resultados aplicando el test de normalidad de Shapiro-Wilk con el mismo nivel de confianza y los mismos valores que la Tabla anterior. Donde podemos observar casos similares a los encontrados aplicando el test de Kolmogorov-Smirnov. Por ejemplo, si vemos los resultados obtenidos por los algoritmos en la instancia A-n34-k5 observamos que se cumple también en este caso la condición de normalidad pero si observamos la Tabla 6 (test de Levene) vemos que no se cumple la homocedasticidad. Analizamos cada una de las instancias y vemos que se deben aplicar test no paramétricos en todos los casos.

	AG	MCMP-SRI	MCMP-SRI-HC	MCMP-SRI- Comb
A-n33-k5	0,33	0,21	0,17	*0,02
A-n33-k6	0,08	*0,01	*0,05	*0,03
A-n34-k5	0,82	0,67	0,08	0,61
A-n37-k6	0,32	*0,02	0,25	0,38
A-n45-k7	0,07	0,48	0,18	0,25
A-n53-k7	0,09	0,86	0,29	*0,01

A-n61-k9	0,19	0,77	0,51	*0,01
A-n62-k8	0,29	0,38	0,30	0,81
A-n64-k9	0,20	0,57	0,81	0,42
A-n80-k10	0,34	0,49	0,51	0,45

Tabla 5: Test de normalidad de Shapiro-Wilk

Instancia	Levene
A-n33-k5	*0,000
A-n33-k6	*0,000
A-n34-k5	*0,001
A-n37-k6	*0,001
A-n45-k7	*0,001
A-n53-k7	0,150
A-n61-k9	*0,001
A-n62-k8	*0,017
A-n64-k9	*0,001
A-n80-k10	*0,271

Tabla 6: Test de Homocedasticidad de Levene (basado en medias)

Seguidamente determinaremos si las diferencias de los resultados obtenidos por los algoritmos para cada una de las instancias son estadísticamente significativas. Teniendo en cuenta que no se cumplen las condiciones para realizar los test paramétricos se aplica entonces el test de Kruskal-Wallis para determinar si existen diferencias significativas entre los algoritmos. Utilizaremos el signo (+) para especificar que existen diferencias significativos entre los resultados obtenidos por los algoritmos y (-) en caso contrario. La Tabla 7 muestra los resultados obtenidos para las instancias utilizadas. Podemos observar que en todos los casos las diferencias entre los resultados obtenidos por los algoritmos son estadísticamente significativas.
Ahora para saber entre los resultados de qué algoritmos existen diferencias estadísticamente significativas aplicamos el test de Tukey. Únicamente se encontraron diferencias estadísticamente significativas entre el AG y los tres restantes, esto significa que podemos afirmar con un nivel de confianza del 95% que los algoritmos MCMP-SRI, MCMP-SRI-HC y MCMP-SRI- Comb tienen un mejor desempeño (en cuanto al porcentaje de error) con respecto al AG.

Instancia	Kruskal-Wallis
A-n33-k5	(+)
A-n33-k6	(+)
A-n34-k5	(+)
A-n37-k6	(+)
A-n45-k7	(+)
A-n53-k7	(+)
A-n61-k9	(+)
A-n62-k8	(+)
A-n64-k9	(+)
A-n80-k10	(+)

Tabla 7: Test de Kruskal-Wallis

Con la idea de analizar ahora el desempeño de los cuatro algoritmos sobre el conjunto de problemas aplicaremos un conjunto de test no paramétricos (para muestras relacionadas).

Los siguientes test se han aplicado utilizando el paquete (en Java) CONTROLTEST que se puede obtener en el sitio web público temático SCI2S - *Statistical Inference in Computational Intelligence and Data Mining*. Para una profundización de los test aplicados a continuación ver [Derrac et al. 2011].
Aplicaremos el Test de Friedman, que se trata de un equivalente no paramétrico del test de ANOVA. Calcula el orden de los resultados observados por algoritmo para cada función, asignando al mejor de ellos el orden 1, y al peor el orden k. Bajo la hipótesis nula, que se forma a partir de suponer que los resultados de los algoritmos son equivalentes y, por tanto, sus rankings son similares.

Algoritmo	Ranking
AG	3,99
MCMP-SRI	1,45
MCMP-SRI-HC	2,59
MCMP-SRI-Comb	1,94

Tabla 8: Rankings promedio de los algoritmos (Friedman)

La Tabla 8 muestra los resultados del estadístico de Friedman (distribuido de acuerdo a X^2 con 3 grados de libertad: 21,99). El valor de *p* calculado por el test de Friedman es: 6,55E - 5, (valor muy pequeño y <= 0,05) esto significa que existen diferencias estadísticamente significativas entre los resultados obtenidos por algoritmos. Pasaremos luego en este caso a aplicar algunos test post-hoc para verificarlo. Podemos observar que el algoritmo MCMP-SRI es el que obtiene el mejor ranking y el peor ranking es el obtenido por el algoritmo genético. Es decir que sobre el conjunto de instancias el algoritmo MCMP-SRI ha tenido un mejor desempeño comparado con los otros 3.
Ahora aplicaremos procedimientos post-hoc de identificación de diferencias. El Test de Friedman no identifica las diferencias existentes entre el mejor algoritmo encontrado (denominado como algoritmo de control) y el resto. Este test se limita a detectar la existencia o no de diferencias en todo el conjunto de resultados. Si las encuentran, es necesario proceder con un procedimiento post-hoc de identificación de diferencias (basados en la distribución normal).
En este caso como el test de Friedman encontró diferencias estadísticamente significativas aplicaremos los procedimientos post-hoc y para ello se debe aproximar el estimador *z* de una distribución normal a partir de la diferencias entre dos rankings (la forma de realizarlo puede encontrarse en [Conover 1999], y [Daniel 1990]). Luego de obtenido el valor *z*, es posible a partir de él calcular el valor *p- no ajustado* correspondiente. La utilidad de estos procedimientos post-hoc radica en que son capaces de calcular el valor p-ajustado, considerando la familia de hipótesis completa para cada pareja de algoritmos comparados [Westfall y Young 1993].
A continuación, presentamos algunos procedimientos útiles para detectar diferencias a posteriori. Existen más alternativas, pero apenas ofrecen alguna diferencia de potencial con respecto a las que se detallan aquí [García et al. 2010].
En la Tabla 9 podemos observar que en este caso el algoritmo de control es MCMP-SRI (el mejor algoritmo) y se lo compara con todos los demás, con un valor de $p = 0,05$. Esta Tabla muestra la aplicación de varios test sobre los resultados obtenidos con el test de Friedman. Las dos primeras columnas identifican el orden y el nombre del algoritmo respectivamente. La columna identificada por *p no ajustado* representa el valor calculado para cada algoritmo. Las posteriores columnas representan el valor *p-ajustado* obtenido por cada test que se ha de comparar.

En la Tabla 9 podemos observar teniendo en cuenta los umbrales y los resultados del valor p-*ajustado* para cada test que todos ellos rechazan la hipótesis nula de igualdad de medias, por lo que significa que según estos test existen diferencias estadísticamente significativas entre el algoritmo genético y el MCMP-SRI. Con respecto a los algoritmos MCMP-SRI-HC y MCMP-SRI-Comb en todos los test aplicados no se puede decir que existan diferencias estadísticamente significativas con respecto a MCMP-SRI.

i	*Algoritmo*	*p no ajustado*	*p Bonf*	*p Holm*	*p Holch*	*p Homm*
1	AG	1,00E-05	3,01E-05	3,01E-05	3,01E-05	3,01E-05
2	MCMP-SRI-HC	0,04	0,04	0,14	0,09	0,09
3	MCMP-SRI-Comb	0,37	0,37	0,39	0,39	0,39

Tabla 9: Valores p-ajustado (FRIEDMAN) para test de Bonferroni-Dunn, Holm, Hochberg y Hommel

Para el análisis de los test se debe tener en cuenta los siguientes umbrales:
El procedimiento de Bonferroni-Dunn rechaza las hipótesis que tienen un valor p <= 0; 016.
El procedimiento de Holm rechaza las hipótesis que tienen un valor p <= 0; 025.
El procedimiento de Hochberg rechaza las hipótesis que tienen un valor p <=0; 016.
El procedimiento de Hommel rechaza las hipótesis que tienen un valor p <=0; 025.
El procedimiento de Holland rechaza las hipótesis que tienen un valor p <=0; 025.
El procedimiento de Rom rechaza las hipótesis que tienen un valor p <=0; 016.
El procedimiento de Finner rechaza las hipótesis que tienen un valor p <=0; 033.
El procedimiento de Li rechaza las hipótesis que tienen un valor p <=0; 032.

Algoritmo	*Bonferroni-Dunn*	*Holm*	*Hochberg*	*Hommel (FRIEDMAN)*
AG	(+)	(+)	(+)	(+)
MCMP-SRI-HC	(-)	(-)	(-)	(-)
MCMP-SRI-Comb	(-)	(-)	(-)	(-)

Tabla 10: Resumen de resultados de Test de Bonferroni-Dunn, Holm, Hochberg y Hommel

La Tabla 10 presenta un resumen de los resultados obtenidos por cada test. Donde (+) se muestra si rechaza la hipótesis nula es decir, que "existen diferencias estadísticamente significativas entre los resultados obtenidos por los algoritmos" y (-) representa la aceptación de la hipótesis nula, es decir "que las diferencias no son estadísticamente significativas."

i	*Algoritmo*	*p no ajustado p*	*p Holl*	*p Rom*	*p Finn*	*p Li*
1	AG	1,00E-05	3,01E-05	3,01E-05	3,01E-05	1,63E-05
2	MCMP-SRI-HC	0,04	0,09	0,091	0,09	0,07
3	MCMP-SRI-Comb	0,39	0,39	0,39	0,39	0,39

Tabla 11: Valores p-ajustado (FRIEDMAN) para test de Holland, Rom, Finner y Li

La Tabla 11 muestra información similar a la Tabla 9 pero en este caso muestra los valores de *p ajustado* para los test de Holland, Rom, Finner y Li. Podemos observar un resultado idéntico al obtenido por los test de la Tabla 9 y esto se ve reflejado en la Tabla resumen 12.

Algoritmo	*Holland*	*Rom*	*Finner*	*Li*

AG	(+)	(+)	(+)	(+)
MCMP-SRI-HC	(-)	(-)	(-)	(-)
MCMP-SRI-Comb	(-)	(-)	(-)	(-)

Tabla 12: Resumen de resultados de Test de Holland, Rom, Finner y Li (FRIEDMAN)

Observando el resumen de las Tablas 10 y 12 podemos decir que los resultados obtenidos por el algoritmo MCMP-SRI tienen diferencias estadísticamente significativas con los resultados obtenidos por el AG para todos los test aplicados. En cuanto a los resultados obtenidos por los algoritmos MCMP-SRI-HC y MCMP-SRI-Comb para todos los test aplicados no se puede decir que las diferencias sean estadísticamente significativas con respecto a los resultados obtenidos por el algoritmo MCMP-SRI.

5. CONCLUSIONES

Los algoritmos evolutivos son algoritmos de búsqueda robustos en el sentido que proporcionan soluciones buenas en una amplia clase de los problemas que de otra manera serían computacionalmente intratables. Para mejorar el funcionamiento de los AEs, los enfoques multirecombinativos permiten múltiples intercambios de material genético entre múltiples padres y con ello mejoran la velocidad de convergencia. Para mejorar el proceso de búsqueda, mediante un mejor equilibrio entre la exploración y la explotación, el concepto de *stud* e inmigrantes aleatorios fue insertado en MCMP-SRI. La presencia del *stud* asegura la retención de los buenos rasgos de soluciones anteriores y los inmigrantes aleatorios, como una fuente continua de diversidad genética, evita la convergencia prematura.

El Problema de Ruteo de Vehículos con Capacidad limitada (CVRP), es una de las variantes del VRP que es considerado emblemático en el campo de la distribución, logística y tráfico.

En este trabajo hemos presentado dos versiones de MCMP-SRI combinadas con Hill-Climbing (MCMP-SRI-HC y MCMP-SRI-Comb) además hemos presentado una familia de test no paramétricos para comparar los resultados (error porcentual) de los algoritmos. Una vez determinada la necesidad de aplicar test no paramétricos se analizaron los resultados realizando comparaciones múltiples aplicando test apareados y no-apareados. Finalmente, se aplicaron diversos test post-hoc.

En cuanto a los resultados obtenidos, luego de aplicar test no-apareado (Kruskal-Wallis) podemos afirmar que existen diferencias estadísticamente significativas entre los algoritmos en cada uno de los problemas tratados. Al aplicar el test post-hoc (Tukey) confirmamos que las diferencias en cada uno de los problemas estaban entre los resultados obtenidos por el algoritmo AG con respecto a resultados obtenidos por los otros tres algoritmos (MCMP-SRI, MCMP-SRI-HC y MCMP-SRI-Comb).

Seguidamente aplicamos un test apareado (Friedman). En Friedman obtuvimos diferencias estadísticamente significativas en los resultados de los algoritmos. Por esta razón aplicamos una serie de test post-hoc y pudimos comprobar que los resultados eran similares ya que en todos los casos se obtuvo que los resultados del algoritmo MCMP-SRI tienen diferencias estadísticamente significativas con respecto a los resultados obtenidos por el AG. Si bien los algoritmos propuestos obtienen resultados similares a MCMP-SRI es importante destacar el fuerte estudio estadístico realizado sobre los resultados.

Trabajos futuros incluirán otras formas de auto-adaptación e hibridación para otras variantes de VRP.

6. AGRADECIMIENTOS

Se agradece la cooperación del equipo de proyecto del LabTEm y a la Universidad Nacional de la Patagonia Austral, de los cuales se recibe apoyo continuo.

7. REFERENCIAS

[Bäck 1996] T. Bäck. Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, genetic Algorithms. Oxford University Press, 1996.
[Bäck et al. 1997] T. Bäck, D.B. Fogel, and Z. Michalewicz, editors. Handbook of Evolutionary Computation. Oxford University Press, 1997.
[Blum y Roli 2003] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys, 35(3):268–308, 2003.
[Conover 1999], W. Conover, 1999 practical nonparametric statistics.
[Cook et al. 1998] W.J. Cook, W.H. Cunningham, W.R. Pulleyblank, and A. Schrijver. Combinatorial Optimization. Wiley Interscience, 1998.
[Crainic y Toulouse 2003] T. Crainic and M. Toulouse. Parallel strategies for meta-heuristics. En: Glover, F. y Kochenberger, G. (Eds.), Handbook of metaheuristics, Kluwer academic publisher, 2003.
[Daniel 1990] W.W. Daniel. Applied nonparametric statistics. 1990. PWS-Kent Devon, UK.
[Dantzig y Ramser 1959] G. B Dantzig, R.H. Ramser: "The Truck Dispatching Problem". Management Science 6, 80–91. (1959).
[Derrac et al. 2011] J. Derrac, S. García, D. Molina, and F. Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation, 2011.
[De San Pedro et al. 2001] M.E. De San Pedro, D. Pandolfi, A. Villagra, G. Vilanova, R. Gallard 2001; Stud and immigrants in multirecombined evolutionary algorithm to face weighted tardiness scheduling problems; CACIC'01 VII Congreso Argentino de Ciencias de la Computación, El Calafate, Argentina, octubre, pp. 1251-1258.
[De San Pedro et al. 2003] M.E. De San Pedro, M. Lasso, A. Villagra, D. Pandolfi, R. Gallard, 2003; Solutions to the Dynamic Average Tardiness Problem in the Single machine Environments; CACIC'03 IX Congreso Argentino de Ciencias de la Computación, La Plata, Argentina, pp. 1251-1258.
[Dorigo 1992] M. Dorigo. Optimization, Learning and Natural Algorithms. PhD thesis, Dipartimento di Elettronica, Politecnica di Milano, 1992.
[Dorigo y Di Caro 1999] M. Dorigo and G. Di Caro. The ant colony optimization metaheuristic. In D. Corne, M. Dorigo, and F. Glover, editors, New Ideas in Optimization, pages 11–32. McGraw Hill, 1999.
[Eberhart y Kennedy 1995] R. Eberhart and J. Kennedy. A new optimizer using particles swarm theory. In Sixth International Symposium on Micro Machine and Human Science (Nagoya, Japan), IEEE Service Center, Piscataway, pages 39–43, 1995.
[Feo y Resende 1995] T.A. Feo and M.G.C. Resende. Greedy randomized adaptive search procedures. Journal of Global Optimization, 6:109–133, 1995.
[García et al. 2010] S. García, A. Fernández, J. Luengo, and F. Herrera. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. Information Sciences, 180(10):2044– 2064, 2010.
[Gauch 2002] H.G. Gauch. Scientific Method in Practice. Cambridge, 1st edition, 2002.

[Glover y Kochenberger 2002] F. Glover and G. Kochenberger. Handbook of Metaheuristics. Kluwer Academic Publishers, Norwell, MA, 2002. Computer and Operations Research, 13:533–549, 1986.
[Glover y Laguna 1993] F. Glover and M. Laguna. Tabu search in Modern Heuristic Techniques for Combinatorial Problems. John Wiley y Sons, 1993.
[Goldberg. 1989] D. Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Publishing Co., 1989.
[Hansen y Mladenovic 1997] P. Hansen and N. Mladenovic. Variable neighborhood search for the p-median. Location Science, 5:207–226, 1997.
[Hart et al. 2005] W. Hart, N. Krasnogor, and J. Smith. Recent Advances in Memetic Algorithms. Springer, 2005.
[Hochbaum 1996] D. S. Hochbaum. Approximation Algorithms for NP-Hard Problems. International Thomson Publishing, 1996.[Kelly y Xu 1996] Kelly, J., Xu, J. P.: "A Network Flow-Based Tabu Search Heuristic for the Vehicle Routing Problem". Transportation Science, 30:379-393. (1996)
[Hoos y Stützle 2004] H. Hoos and T. Stützle. Stochastic local search: Foundations y applications. Morgan Kaufmann. 2004.
[Kirkpatrick et al. 1983] S. Kirkpatrick, J. Gelatt, and M. Vecchi. Optimization by simulated annealing. Science, 220:671–680, 1983.
[Korf 1985] R. Korf. Depth-first iterative-deepening: An optimal admissible tree search. Artificial Intelligence, 27(1):97–109, 1985.
[Lasso et al. 2004] M. Lasso, D. Pandolfi, M. E. De San Pedro, A.Villagra, R. Gallard; “Solving Dynamic Tardiness Problems in Single Machine Environments”; Congress on Evolutionary Computation - CEC ’04; Portland, U.S.A, Vol 1, 1143- 1149, (2004)
[Lenstra y Kan 1981] J. Lenstra, A. R. Kan: Complexity of vehicle routing and scheduling problems. Networks 11, 221-227. (1981)
[Papadimitriou y K. Steiglitz 1998] C. Papadimitriou and K. Steiglitz. Combinatorial Optimization. Dover Publications, 1998.
[Louren et al. 2001] H. R. Louren, O. Martin, and T. Stützle. A beginner´s introduction to iterated local search. In MIC, pages 1–6, 2001.
[Pandolfi et al. 2001] D. Pandolfi, G. Vilanova, M.E De San Pedro, A. Villagra; R. Gallard 2001; Solving the single-machine common due date problem via studs and immigrants in evolutionary algorithms; Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, Vol. III Emergent Computing and Virtual Engineering, pp. 409-413, Orlando, Florida.
[Pandolfi et al. 2003] D. Pandolfi, M. Lasso, M.E. De San Pedro, A. Villagra, R. Gallard 2003; Evolutionary Algorithms to solve average tardiness problems in the single machine environments; CSITeA03 International Conference on Computer Science, Software Engineering Information Technology, E-Business and Applications, Rio de Janeiro, Brazil, pp. 444-449.
[Russell y Norvig 1995] S. Russell and P. Norvig. Artificial Intelligence: A Modern Approch. Prentice-Hall, 1995.
[Vazirani 2003] V. Vazirani. Approximation Algorithms. Springer, 2003.
[Villagra et al. 2001] A. Villagra, D. Pandolfi, G. Vilanova, M.E De San Pedro; R. Gallard 2001; Adaptability of multirecombined evolutionary algorithms in the single-machine common due date problem; Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, Vol. III Emergent Computing and Virtual Engineering, pp. 401-404, Orlando, Florida.

[Villagra et al. 2012] A. Villagra, D. Pandolfi, G. Leguizamón, Handling constraints with an evolutionary tool for scheduling oil wells maintenance visits. September 2012. Pages 1-19. doi 10.1080/0305215X.2012.713354.
[Westfall y Young 1993] P.H. Westfall and S.S. Young. Resampling-based multiple testing: Examples and methods for p-value adjustment, volume 279. Wiley-Interscience, 1993.
[Whewell y Butts 1989] W. Whewell and R.E. Butts. Theory of Scientific Method. Hackett Pub Co Inc, 2nd edition, 1989.