ISSN 2007-9737

Efecto de la calibración de parámetros mediante un diseño Taguchi $L_9 3^4$ en el algoritmo GRASP resolviendo el problema de rutas de vehículos con restricciones de tiempo

Alma Danisa Romero-Ocaño[1], M.A. Cosío-León[1], Víctor Manuel Valenzuela-Alcaraz[1], Gener J. Avilés-Rodríguez [1], Anabel Martínez-Vargas[2]

[1] Universidad Autónoma de Baja California-FIAD, Ensenada, Baja California, México

[2] Universidad Politécnica de Pachuca, Zempoala, Hidalgo, México

{danissa.romero, cosio.maria, valenzuela.victor, gener.aviles}@uabc.edu.mx, anabel.martinez@upp.edu.mx

Resumen. Los algoritmos metaheurísticos son procedimientos de tipo caja negra que analizan un subconjunto de soluciones posibles para proponer una solución a un problema o un conjunto de instancias. Previo a su uso se requiere seleccionar un vector de parámetros óptimo, P^*, tarea conocida como calibración de la metaheurística. El vector P^* afecta en la eficiencia de la metaheurística al resolver un problema determinado. En este trabajo se analiza el efecto de calibración de parámetros al usar el procedimiento estadístico Taguchi $L_9 3^4$ en el algoritmo metaheurístico, Procedimiento de Búsqueda basado en Funciones Voraces Aleatorias Adaptativas (por su sigla en inglés, GRASP) para resolver el problema de rutas de vehículos con restricciones de tiempo (VRPTW). La efectividad de P^*, se valida utilizando un subconjunto de instancias propuestas en la literatura. Los resultados ofrecidos por el algoritmo en un subconjunto de instancias de 25 clientes mejoran en promedio a los reportados en la literatura, utilizando el P^* obtenido por la calibración mediante Taguchi.

Palabras clave. Calibración de parámetros, Taguchi, metaheurística, GRASP, VRPTW.

Effect of Parameters Tuned by a Taguchi Design $L_9 3^4$ in the GRASP Algorithm to Solve the Vehicle Routing Problem with Time Windows

Abstract. Metaheuristic algorithms are black box procedures that analyze a subset of possible solutions to solve a problem or a set of instances. Before they are implemented, it is necessary to select an optimum parameter vector P^*, a task known as tuning. The vector P^* affects the efficiency of metaheuristics in solving a given problem. In this paper, the impact of tuning parameters using the Taguchi $L_9 3^4$ statistical procedure is analyzed. The effect of this method is analyzed in the metaheuristic algorithm named Greedy Randomized Adaptive Search Procedure (GRASP), solving the problem of Vehicle Routes with Time Windows (VRPTW). The results offered by the algorithm in a subset of instances of 25 clients improve on average to those reported in the literature, using a P^* proposed by Taguchi calibration.

Keywords. Tuning, Taguchi, metaheuristic, GRASP, VRPTW.

1. Introducción

Los parámetros son los componentes configurables de un algoritmo metaheurístico. Las metaheurísticas son sensibles al valor de sus parámetros [52]. El teorema de *No Free Lunch* (por su sigla en inglés, NFL) [50], establece que no existe un P^* que de solución a todos los problemas de optimización, por lo que no se deben generalizar.

La selección de P^* en los algoritmos metaheurísticos depende del problema y es una

cuestión importante para asegurar la calidad de soluciones [32], por lo que se requiere una estrategia sistemática y robusta del diseño de parámetros para realizar con precisión y eficiencia la calibración [3].

En el campo de las metaheurísticas, los parámetros se fijan: por analogía, donde el trabajo implica la revisión de la literatura, en problemas similares; y de manera empírica, que requiere de tiempo, trabajo intensivo, y un experto en el problema que se aborda [8].

Adenzo et al en [1], explica que, la selección de valores de parámetros se justifica comúnmente en una de las siguientes maneras: (1) La mayoría de los estudios proporcionan una lista de los parámetros y su valor, justificando que se han establecido experimentalmente, sin especificar el tipo de experimento; (2) Otros estudios proporcionan valores de parámetros sin ninguna explicación adicional; (3) En algunos casos los autores muestran valores de parámetros que han sido reportados en otros estudios, con frecuencia, esto se hace sin confirmar que estos valores sean efectivos en el problema que se este estudiando; (4) En menor número de casos, los trabajos informan los diseños experimentales que emplearon para establecer los valores de los parámetros utilizados para las pruebas computacionales.

Existen dos estrategias diferentes para el ajuste de parámetros: 1) El ajuste online que actualiza los parámetros adaptativamente. En los enfoques adaptativos, los parámetros se codifican en la representación de soluciones y como resultado, al ir modificando la solución, el valor de los parámetros se cambiará. 2) El ajuste offline, hace referencia al establecimiento de parámetros antes de la ejecución de las metaheurísticas.

En esta estrategia para establecer los parámetros, una técnica que se utiliza y que se explora en este trabajo es el diseño de experimentos (DOE) [52] y las mejores prácticas asociadas con la aplicación de técnicas de DOE para las pruebas computacionales se pueden encontrar en [13, 7].

El DOE se define como el proceso de planificación de un experimento para recopilar los datos apropiados y después analizarlos mediante métodos estadísticos para presentar conclusiones válidas y objetivas. Así, hay dos aspectos de cualquier problema experimental: el diseño del experimento y el análisis estadístico de los datos. Estos dos temas están estrechamente relacionados, ya que el método de análisis depende directamente del diseño empleado.

En la aplicación de un DOE se puede utilizar un diseño factorial que estudia el efecto individual y de interacción de varios factores sobre una o varias respuestas. Si el diseño factorial es completo se ejecutan aleatoriamente todas las posibles combinaciones que se pueden formar con los niveles de los factores a investigar. Sin embargo en la práctica a veces no es conveniente realizar tantos experimentos y se utiliza el diseño factorial fraccionado que son aquellos en los que se elige una parte o fracción de los tratamientos de un factor completo. Otra forma de aplicar un DOE es el propuesto por Taguchi, donde define un conjunto especial de arreglos ortogonales para establecer experimentos asociados con la mejora de la calidad en los procesos de fabricación [37].

Discusiones sobre el diseño experimental y el análisis estadístico de los estudios computacionales se pueden encontrar en [7, 15, 22, 24]. Una aplicación del diseño de experimentos a los algoritmos de optimización de red se presenta en [4]. En [5], también utilizan técnicas de diseño experimental para comparar métodos de solución alternativa en el problema de asignación generalizada. En [13], aplican el diseño de experimentos y el descenso gradiente para encontrar valores de parámetros efectivos para dos heurísticas de rutas de vehículos.

Referencias de ajuste adicionales son: [49] en una aplicación en problemas de rutas de vehículos y [51] en un procedimiento de búsqueda tabú para un problema de diseño de redes de telecomunicaciones.

Algunos trabajos en la literatura, consideran en su etapa de diseño del experimento incluir el proceso de ajuste de parámetros por medio de procedimientos automatizados [12, 26, 8, 21, 2], es decir, algoritmos diseñados para la búsqueda del vector P^*, [25, 34]. Algunos autores han abordado la selección del vector de P^* apoyándose en el uso de un diseño experimental, ya sea un diseño

factorial completo o fraccionario acorde al número de parámetros.

En [13], se propone una combinación de diseño de experimentos factorial completo o fraccionado y descenso de gradiente para encontrar el vector P^*. El proceso lo divide en dos etapas: busca una buena configuración para cada instancia del conjunto de pruebas del problema, utilizando un DOE, para después combinar las configuraciones calculando el valor promedio de cada parámetro entre todas las configuraciones. Este procedimiento solo es aplicable a parámetros numéricos.

Un proceso similar es empleado por [1], en el sistema que denominaron CALIBRA. Este evalúa cada configuración con un diseño factorial completo de 2 niveles (2 valores por parámetro). Iterativamente, explora el espacio de las configuraciones en busca de regiones prometedoras usando un DOE fraccionario, basado en arreglos ortogonales de Taguchi, evalúan 9 configuraciones alrededor de la mejor configuración encontrada hasta el momento. Este método tiene dos desventajas: no acepta parámetros continuos y puede optimizar un máximo de 5 parámetros.

En [8], se propone un procedimiento bajo el nombre de F-Race, basado en algoritmos de selección de la mejor configuración para determinar parámetros de metaheurísticas. F-Race, inspirado en algoritmos de aprendizaje de máquinas, propone la evaluación de un conjunto de configuraciones candidatas, donde se descartan las peores tan pronto como se tenga suficiente evidencia estadística en su contra. La gran desventaja de F-Race es la utilización de un diseño factorial completo, sobre el espacio de las configuraciones, pues el conjunto de configuraciones candidatas crece exponencialmente respecto a la cantidad de parámetros.

ParamILS, desarrollado por [25], consiste en una plataforma (framework), para resolver el problema de selección de configuraciones, presentan métodos para optimizar el rendimiento de un algoritmo, en instancias de prueba, variando el conjunto de parámetros ordinales y/o categóricos que se aplican en el algoritmo. Esta propuesta se basa en un algoritmo de búsqueda local iterada, *primero el mejor*, que utiliza una combinación de configuraciones seleccionadas aleatoriamente, en su inicialización; y cierto número de movimientos aleatorios para perturbar la soluciónes, con el fin de evitar mínimos locales.

Es de interés en esta investigación conocer el efecto de un procedimiento de calibración automatizada en los resultados que una metaheuristica ofrece. En particular, éste trabajo se enfoca en el algoritmo GRASP [19], para el problema VRPTW [41]. Comparado con otras metaheurísticas, GRASP ofrece varias ventajas para abordar el VRPTW: ajuste de pocos parámetros (tamaño de lista de candidatos y el criterio de parada), calidad de soluciones y fácil implementación [11, 31, 42]. Al algoritmo basado en GRASP para resolver el problema de VRPTW se le identificará como G-VRPTW.

La aportación principal de este trabajo, es mostrar el efecto de la calibración manual y el DOE Taguchi para la selección del vector P^* en G-VRPTW en las soluciones que éste genera, y para mostrarlo este artículo está organizado como sigue: Sección 2 describe el algoritmo metaheurístico GRASP, Sección 3 muestra el DOE Taguchi, Sección 4 describe el problema VRPTW, Sección 5 describe las instancias de Solomon, Sección 6 muestra la metodología, Sección 7 presenta el proceso de experimentación, Sección 8 describe los resultados, Sección 9 presenta una discusión sobre los hallazgos y finalmente, Sección 10 presenta las conclusiones y el trabajo futuro en la investigación.

2. Algoritmo metaheurístico GRASP

Fue desarrollado a finales de la década de 1980 por Feo y Resende, [20], para estudiar problemas de alta complejidad combinatoria. [39]. Tiene dos fases, una de construcción y una de búsqueda local [20, 44]. La primera fase consiste en construir iterativamente una solución factible incorporando un cliente a la vez. Cada cliente que se va agregando a la solución es determinado por una función voraz, la cual mide el beneficio o costo de agregarlo a la solución que se construye.

La función voraz construye una lista restringida de candidatos (LRC), formada por un subconjunto de clientes, aquellos que al incluirse a la solución parcial generan un menor costo (este

es el aspecto voraz del algoritmo). Los clientes son seleccionados aleatoriamente y agregados a la solución parcial. Una vez seleccionados se eliminan, enseguida se actualiza la LRC y se calcula el costo de la solución. Estos procesos se repiten hasta agotar todos los clientes.

La LRC es determinada de acuerdo con la Ecuación 1, donde *LC* es el total de clientes y α es un parámetro que toma valores entre 0 y 1 para seleccionar de manera aleatoria al cliente que se irá agregando a la solución. Un α=0 vuelve al algoritmo completamente voraz, mientras que un α=1 lo hace totalmente aleatorio. En general α sirve para establecer un balance entre costo computacional y calidad de las soluciones, es decir, entre más voraz sea la fase de construcción, mayor será el tiempo de ejecución [44]:

$$LRC = LC * \alpha. \tag{1}$$

La segunda fase de GRASP consiste en mejorar la solución generada en la etapa de construcción. El uso de algoritmos de búsqueda en vecindarios es lo más común para este propósito. Se realiza de forma iterativa remplazando sucesivamente la solución actual por otra que se encuentre en el conjunto de soluciones vecinas.

Cada vez que se ejecutan las dos etapas, la solución obtenida se almacena y se procede a realizar una nueva iteración, guardando la mejor solución que se haya encontrado hasta el momento. El GRASP se presenta en el Algoritmo 1.

Algoritmo 1: Forma general del algoritmo GRASP

D= instancia de prueba, α, condición de parada;
mientras *(no se cumpla la condición de parada)* **hacer**
    S= Algoritmo Constructivo (*D*);
    S= Búsqueda local (*S*);
    Registrar mejor solución (*S*, *MS*);
Regresar la mejor solución (*MS*);
Fin *GRASP*;
D= datos de entrada, *S*= solución, *MS* = mejor solución.

3. DOE Taguchi $L_9 3^3$

El DOE propuesto por Taguchi identifica las mejores configuraciones de parámetros y mejora la calidad de las soluciones con la técnica de matriz ortogonal y la relacion señal-ruido (S/R) [3]. Tiene como objetivo minimizar la variabilidad en la operación, seleccionando las combinaciones más adecuadas de los niveles de factores controlables en comparación con los factores incontrolables. Implica la maximización del rendimiento y la calidad a un menor costo de operación [40].

En el DOE Taguchi, las características de calidad se presentan en tres escenarios: lo más bajo es mejor, lo nominal es mejor y entre más alto mejor. En el caso de éste trabajo el objetivo es minimizar la distancia, por lo que se utilizó lo más bajo es mejor, aplicando la Ecuación 2, [40]:

$$S/R = -10Log(1/n \sum_{i=1}^{n} Y_i{}^2), \tag{2}$$

donde S/R es la razón señal a ruido, n es el número de configuraciones, i es la configuación, Y_i es la media de los datos de la configuración i. En el DOE Taguchi se debe cumplir los pasos que se describen en [33, 40], los cuales son explicados en la sección de experimentación.

4. Descripción del problema en VRPTW

En la actualidad las organizaciones buscan optimizar sus procesos, mercados nuevos y oportunidades de crecimiento. La mejora en los servicios de transporte y distribución es uno de sus principales desafíos cuando se interesan en el diseño adecuado de la cadena de abastecimiento, ya que esto les permitirá reducir costos, cumplir con los requerimientos de sus clientes y evaluar la posibilidad de ampliar mercados [9].

El objetivo de los problemas de ruta de vehículos (VRP) [17], es entregar bienes a un conjunto de clientes con demandas conocidas, a un costo mínimo, encontrando las rutas óptimas que inician y terminan en el almacén o depósito. Cada cliente debe ser visitado una sola vez, satisfaciendo su demanda y no excediendo la capacidad máxima

ISSN 2007-9737

de carga de los vehículos [41, 10]. El VRPTW es una variante del VRP, donde clientes ubicados en una zona geográfica deben ser visitados en un tiempo determinado por una flota de vehículos con capacidad de carga limitada, que parten de un depósito y hacen un recorrido específico.

Se considera que cada uno de los clientes imponen una restricción temporal, llamada ventana de tiempo, que refiere al tiempo de servicio o entrega de algún producto [48]. Considerando las dos etapas del GRASP, a continuación se describe el desarrollo para la solución del problema G-VRPTW.

4.1. Fase de construcción

De forma general, el algoritmo G-VRPTW construye soluciones en paralelo, es decir varias rutas a la vez bajo los siguientes criterios:

1. El primer cliente seleccionado es conectado al depósito. A partir del segundo cliente seleccionado de la LRC se evalúa el agregarlo al final de las rutas construídas, si cumple con la restricción de capacidad de carga de los vehículos, se agrega a un subconjunto de rutas de interés; en caso contrario se inicia una nueva ruta desde el depósito.

2. El subconjunto de rutas resultante, son evaluadas respecto al tiempo. Se determina el tiempo acumulado hasta el último cliente de cada una de ellas, más el tiempo de recorrer la distancia de ese cliente al cliente seleccionado. La suma anterior debe ser menor o igual al tiempo de la ventana de salida del cliente seleccionado. En caso de incumplir al agregar este último servicio a las rutas disponibles, se genera una nueva ruta conectando el cliente seleccionado al depósito.

3. El cliente seleccionado, es insertado al final de la ruta que haya cumplido todas las restricciones anteriores y proporcione el menor tiempo de recorrer desde el último cliente de la ruta al cliente seleccionado (distancia). Considerar que: si el vehículo llega antes del inicio de la ventana de servicio, debe esperar. Este proceso se repite hasta agotar la LRC, otorgando como solución final aquella que de el menor tiempo de recorrido total.

4.2. Fase de mejora

El algoritmo G-VRPTW aplica las siguientes búsquedas locales: primero hace un recorrido para identificar las rutas que quedaron construidas con un solo cliente y evalua insertarlo en otra existente. En segundo lugar hace un recorrido por los clientes evaluando si, al reubicarlo al final de la ruta, mejora la solución. La tercer búsqueda local evalúa la posibilidad de destruir una ruta al reubicar a los clientes de la misma en otra ruta. Antes de aplicar estos cambios se evalúa primero si se cumplen con las restricciones del VRPTW y minimizan la distancia en la ruta.

5. Instancias de Solomon

El conjunto de instancias del problema VRPTW propuesto por Solomon [46] es punto de referencia en la validación de técnicas metaheurísticas para propuestas de solución al VRPTW, en [47] se muestran los mejores encontrados. El conjunto de pruebas consta de 168 instancias, las cuales se presenta con 25, 50 y 100 clientes, con las siguientes capacidades de carga de los vehículos y nomenclaturas:

- Las que tienen una capacidad de carga de 200 son identificadas con C1(9), R1(12) y RC1(8).
- Con capacidad de 700, C2(8).
- Y el grupo con capacidad de 1000 tienen la nomenclatura R2(11) y RC2(8).

El número entre paréntesis indica cantidad de instancias en el grupo. Los grupos de instancias difieren entre sí respecto al ancho de las ventanas de tiempo; como se muestra en la Tabla 1. Algunos clientes tienen ventanas de tiempo estrecho, identificada con 1, mientras que otros clientes tienen ventanas de tiempo amplias, identificadas con el número 2. Existen diferencias de igual forma en la ubicación de los clientes según sus coordenadas y densidad de ventanas.

En la Tabla 1, se identifica geográficamente a los clientes como:

- *C* = *Cluster*, es decir distribuidos por grupos.
- *R* = *Random*, distribuidos aleatoriamente.
- *RC* = Distribuidos de forma mixta, es decir en grupo y aleatorios.

El conjunto de los problemas 1: R1, C1, y RC1 son de corto horizonte de programación (se generan rutas cortas, entre 5 y 10 clientes). Los problemas 2: R2, C2, RC2 son de largo horizonte de programación, es decir que muchos clientes pueden ser servidos por un mismo vehículo (hasta 30 clientes) generando menos rutas que el conjunto de problemas 1. Por ejemplo, si se selecciona una instancia del conjunto RC1, se hace referencia a que los clientes de esa instancia están distribuídos de forma mixta, tanto en grupos como aleatorios (RC) además sus ventanas son estrechas y el horizonte de programación es corto, indicados ambos por el número 1 en la nomenclatura de la instancia.

Tabla 1. Clasificación de instancias

Clasificación	Características		
Distribución geográfica	C	R	RC
Densidad de ventana	25 %	50 %	75 %
Ancho de ventana	1	2	
Horizonte de programación	1	2	

Hutter et al. [25] y Birattari et al. [8] mencionan que usar más o menos instancias de prueba en un problema específico, no tiene una respuesta directa, ya que un conjunto de instancias demasiado pequeño, podría conducir a un buen desempeño del algoritmo en el subconjunto evaluado, pero ocasionaría un pobre desempeño en las instancias no evaluadas, por otro lado, un conjunto de instancias muy grande, incrementaría el tiempo de ejecución del proceso de búsqueda, proporcionando un rendimiento promedio del algoritmo en ellas.

6. Metodología

En la Figura 1, se muestra el proceso de un algoritmo metaheurístico, iniciando con un conjunto de valores y P^* que entran a un proceso de calibración para generar el vector de parámetros del algoritmo que resuelve un conjunto de instancias, regresando una solución de buena calidad. Este vector es utilizado para evaluar la calidad de la solución al utilizar los parámetros calibrados [18].

La calibración del G-VRPTW se realizó con el proceso del DOE Taguchi, se aplicó al calibrar el 20 % de las instancias de Solomon, seleccionadas de forma aleatoria. El 80 % restante, se ejecutó con el vector P^* generado una vez aplicado el DOE Taguchi, con el propósito de validar su efectividad.

La cantidad de instancias a utilizar para calibración y para pruebas se determinó de acuerdo al método de *hold-out test* [28] y el principio de pareto (20/80), [30]. Los resultados del algoritmo al utilizar el vector P^*, se compararon con los promedios de los resultados que se obtuvieron de 10 ejecuciones del algoritmo G-VRPTW, donde el vector de P^* fué establecido mediante calibración manual (resultados que identificaremos como **Fase 1**).

7. Experimentación

A continuación, se detallan los pasos del DOE Taguchi [33, 40]:

Paso 1. Formulación del problema. El objetivo de G-VRPTW es minimizar distancia, el modelo matemático del problema VRPTW se explica en [48].

Paso 2. Identificar las características de rendimiento de salida relevantes para el problema. Se consideran los resultados obtenidos al ejecutar el algoritmo en diferentes pruebas pilotos, modificando los parámetros del mismo de una forma empírica, concentrando los datos finales en la Tabla 2.

El porcentaje del Gap es la métrica utilizada en el presente trabajo, se calcula con la Ecuación 3 y se define como la variación porcentual de la solución encontrada por el algoritmo Sol_{Alg} a una de las siguientes soluciones: (a) solución inferior

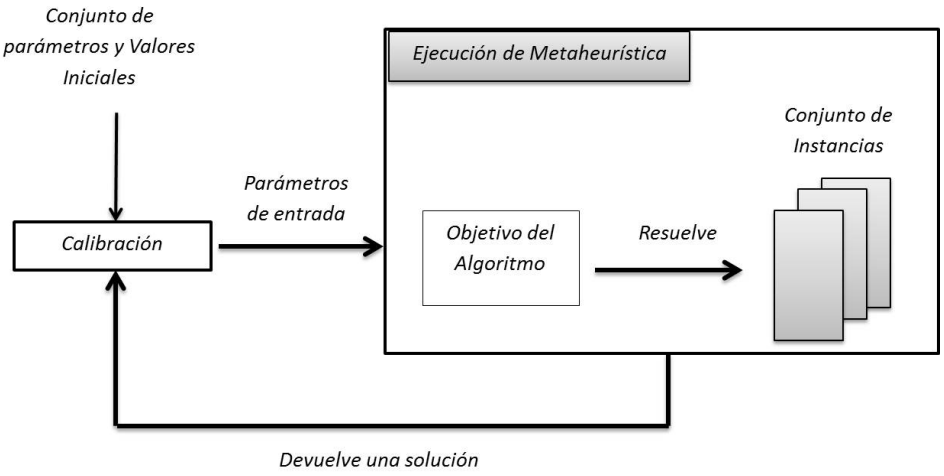

Fig. 1. Proceso de ejecución de un algoritmo metaheurístico

Tabla 2. Resultados del % Gap del 80 % de las instancias obtenido en la **Fase 1**

Grupo de Instancias	Num. de Clientes		
	25	50	100
C1	1.4	8.6	25.8
C2	10.0	23.1	33.1
R1	2.5	7.2	13.3
R2	6.1	27.9	36.9
RC1	0.1	7.6	31.9
RC2	6.1	27.9	36.9
Promedio	5.1	16.5	29.3

(superior), (b) solución óptima, y (c) solución mejor conocida, como se describe en la Figura 2. En este trabajo la calidad de las soluciones se mide con respecto a la solución mejor conocida Sol_{MC} [18, 53]:

$$\%Gap = \frac{Sol_{alg} - Sol_{MC}}{Sol_{MC}} * 100, \qquad (3)$$

Al analizar los datos de la **Fase 1**, es evidente la necesidad de explorar otra forma de calibración de parámetros en la busqueda de mejores resultados.

Paso 3. Identificar los factores de control. Los factores de control son aquellos que afectan la solución del problema. En la metodología del GRASP se presentan tres (parámetros):

- Ordenamiento, parámetro cualitativo.
- Valor de α, parámetro cuantitativo.
- Número de iteraciones, parámetro cuantitativo.

Paso 4. Seleccionar los niveles de los factores. Los niveles que se establecieron para los parámetros o factores que influyen en la solución final del algoritmo, fueron elegidos bajo los siguientes criterios:

Factor 1. Ordenamiento. Este se refiere a la forma de organizar la lista de candidatos en la *fase de construcción* de las rutas.

Niveles:

Nivel 1. Los clientes, según su tiempo de inicio de atención son ordenados de menor a mayor [a], es decir, las rutas se construyen considerando el tiempo.

Nivel 2. Los clientes son ordenados de menor a mayor distancia respecto al depósito [dd], esto es, las rutas se construyen considerando la distancia.

Nivel 3. Mixto. En éste ordenamiento se hizo una combinación de los dos anteriores, considerando: el primer 20 % de clientes se ordenan como el Nivel 2, el siguiente 60 % como el Nivel 1 y el 20 % restante como el Nivel 2. Este orden es con el fin de iniciar y terminar la construcción de las rutas con los clientes más cercanos al depósito y en la

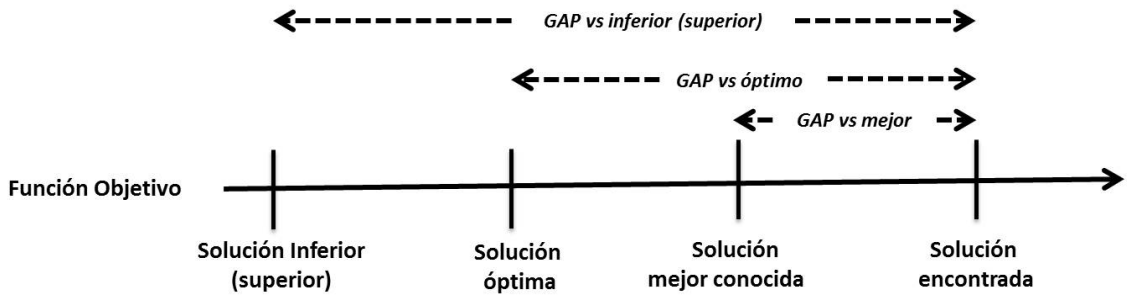

Fig. 2. Evaluación del desempeño respecto a la calidad de la solución en un problema de minimización

fase intermedia de la construcción de las rutas ir considerando a los clientes que solicitan el servicio más temprano.

Factor 2. Valor de α. Factor que determina qué tan voraz es la metaheurística G-VRPTW.

Niveles:

Nivel 1, α = Variable. Se considera el valor de α variable durante la ejecución del algoritmo. Esto se refiere a que se inicia con un α =3/n, donde n es el total de clientes, este valor se mantiene hasta que se hayan atendido el 50 % del total de clientes. Para el 25 % de los clientes siguientes se utiliza un α =2/n y finalmente para el 25 % del total de clientes restantes se utiliza un α =1/n. Por ejemplo para un problema con 100 clientes: un α = 0.03 será utilizado cuando se disponga de 100 a 50 clientes, un α = 0.02 se utilizará cuando queden de 49 a 25 clientes y un α = 0.01 cuando queden menos de 25 clientes. Lo anterior deja el valor de α en función del número de clientes, considerando como *LRC* la lista restringida de candidatos, calculada con la Ecuación 1.

Nivel 2, α= 0.90. Al aumentar el valor de α se está introduciendo variabilidad al momento de seleccionar aleatoriamente al cliente, ya que la LRC será mayor y esto ayudará al proceso para evitar caer en óptimos locales.

Nivel 3, α = 3/n, es decir, utilizar un valor fijo de 3 clientes para la LRC. El considerar la selección constante en 3 clientes se determinó con base en la literatura [31], [11].

Factor 3. Iteraciones. El número de veces que iterará el algoritmo buscando mejorar la solución guardada en memoria.

Niveles:

Nivel 1 = 50,000. Establecidas en las mismas condiciones de la Fase 1 y en el resto, con incrementos de igual magnitud a esta base.

Nivel 2 = 100,000.

Nivel 3 = 150,000.

El concentrado de los factores con sus niveles se aprecian en el Tabla 3.

Tabla 3. Factores seleccionados

Factor	Nivel 1	Nivel 2	Nivel 3
Ordenamiento	a <a	dd <dd	mixto
Alfa	Variable	0.90	3 clientes
Iteraciones	50000	100000	150000

Paso 5. Diseñar el arreglo ortogonal adecuado al problema. Los arreglos ortogonales propuestos por Taguchi son diseños que tienen la propiedad de ortogonalidad, misma que también poseen los diseños factoriales. Estos arreglos son diseños factoriales completos, fraccionados o mixtos, dependiendo del número de factores a estudiar en un caso particular.

Taguchi desarrolló una serie de arreglos que denominó: $L_a(b)^C$, donde: a = Representa el número de pruebas o condiciones experimentales que se deben realizar. Esto es, el número de renglones o líneas en el arreglo. b = Representa los diferentes niveles a los que se tomará cada factor. C = Es el número de efectos independientes que se pueden analizar, esto es, el número de columnas en el arreglo. Se seleccionó un arreglo ortogonal $L_9 3^{4-2}$ de los propuestos por Taguchi que permite estudiar máximo cuatro factores a tres niveles cada uno, ver Tabla 4 [40].

Tabla 4. Arreglo L_9 propuesto por Taguchi [40]

Arreglo $L_9 3^{(4-2)}$				
Configuración	Número de columna			
	1	2	3	4
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2	2	3	1
6	2	3	1	2
7	3	1	3	2
8	3	2	1	3
9	3	3	2	1
2 factores: columnas 1, 2.				
3 factores: columnas 1, 2, 3.				
4 factores: columnas 1, 2, 3, 4.				

De acuerdo a los factores y niveles definidos en el experimento del presente trabajo, ver Tabla 3, solo se ocupan las primeras 3 columnas, correspondientes al factor ordenamiento, factor α y factor iteraciones, respectivamente, quedando así, un arreglo ortogonal $L_9 3^3$.

Paso 6. Ejecutar el experimento y recopilar los datos necesarios. Se ejecutaron los experimentos de acuerdo al arreglo ortogonal, considerando el 20 % de las instancias de Solomon [46], las cuales fueron seleccionadas aleatoriamente. Se realizaron 10 réplicas por cada uno de los arreglos tal como se muestra en la Tabla 5. El concentrado de los resultados al ejecutar el G-VRPTW en el 20 % de las instancias se puede observar en la Tabla 6.

Tabla 5. Concentrado de Factores y Niveles en Aplicación del DOE Taguchi

Config.	*Orden*	*N*	α	*N*	*Iteraciones*	*N*
1	a<a	1	Variable	1	50,000	1
2	a<a	1	0.90	2	100,000	2
3	a<a	1	3 clientes	3	150,000	3
4	dd<dd	2	Variable	1	100,000	2
5	dd<dd	2	0.90	2	150,000	3
6	dd<dd	2	3 clientes	3	50,000	1
7	mixto	3	Variable	1	150,000	3
8	mixto	3	0.90	2	50,000	1
9	mixto	3	3 clientes	3	100,000	2

8. Resultados

Paso 7. Realizar el análisis estadístico e interpretar los resultados.

La normalidad de los datos fué evaluada con el estadístico Anderson - Darling [6], como se muestra en la Figura 3, indicando que los datos siguen una distribución normal, ya que el indicador de *valor p* es de 0.451, siendo mayor que el nivel de significancia α de 0.05.

Tabla 6. Promedios del $\%$ Gap en el 20 % de las instancias

Instancias de prueba 25 clientes						
Config.	***C1***	***C2***	***R1***	***R2***	***RC1***	***RC2***
1	3.78	17.59	0.03	10.26	0.44	8.03
2	0.68	2.17	**-0.63**	3.25	**-0.56**	0.22
3	3.53	53.52	0.008	8.25	0.22	11.63
4	0.90	58.90	0.89	3.07	3.30	5.25
5	0.55	36.97	**-1.44**	1.84	**-3.46**	0.72
6	1.49	13.80	1.27	1.87	3.30	7.68
7	1.27	71.01	0.58	2.82	3.30	7.89
8	0.90	0.62	**-0.25**	2.66	**-1.42**	0.35
9	1.17	57.36	**-0.92**	1.85	2.34	5.75
Instancias de prueba 50 clientes						
1	16.14	23.67	7.97	25.85	2.47	26.72
2	13.42	6.26	8.52	19.96	3.31	16.5
3	13.57	10.20	5.88	22.09	1.71	24.33
4	7.34	14.25	11.72	17.86	6.55	17.79
5	10.72	6.41	7.46	18.42	1.70	15.9
6	8.33	31.97	10.90	21.07	6.70	17.6
7	6.13	8.93	11.06	18.48	7.03	20.20
8	16.80	13.96	8.70	19.62	4.20	14.50
9	5.63	24.85	9.85	20.63	6.84	19.93
Instancias de prueba 100 clientes						
1	25.16	85.27		35.05		13.40
2	41.95	62.98		18.83		19.81
3	26.82	66.16		27.38		19.85
4	29.21	69.72		27.46		22.48
5	45.41	62.63		22.97		21.99
6	27.58	3.12		23.77		12.77
7	32.46	3.64		21.47		17.83
8	41.56	5.32		20.27		19.27
9	17.84	2.36		18.88		14.43

Al hacer el análisis de los resultados ofrecidos por el DOE Taguchi, los datos obtenidos de medias y *S/R*, se muestran en el Tabla 7, utilizados posteriormente para generar las Figuras 4 y 5.

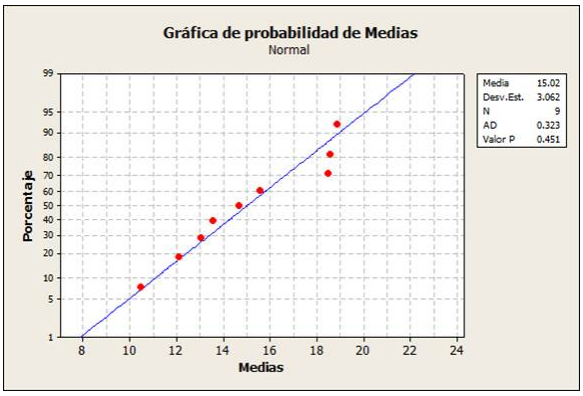

Fig. 3. Gráfica de Evaluación de la Normalidad de Datos

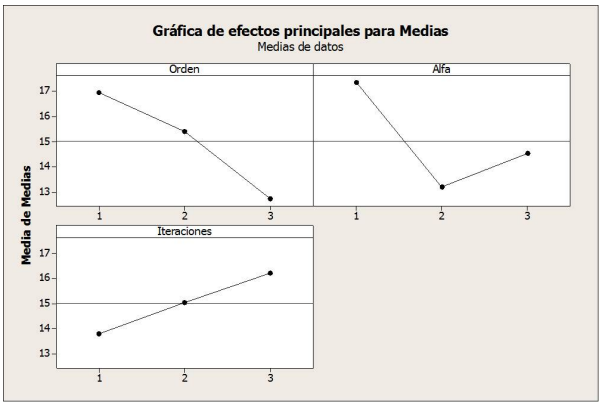

Fig. 4. Gráfica de Efecto Principal para Medias

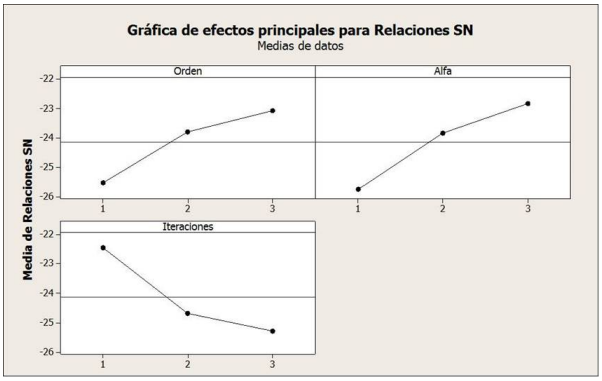

Fig. 5. Gráfica de Efecto Principal para S/R

La combinación más robusta de los niveles de los factores controlables es la que maximiza la razón *S/R*. Los valores de *S/R* se calcularon usando la fórmula lo más bajo es lo mejor, tal como se menciona anteriormente, ya que se desea calcular la longitud total de la ruta más corta para el G-VRPTW.

Tabla 7. Resultados de medias y señal a ruido (*S/R*)

Configuración	*Medias*	*S/R*
1	18.86	-26.29
2	13.55	-24.80
3	18.45	-25.46
4	18.54	-26.06
5	15.55	-25.49
6	12.08	-19.83
7	14.63	-24.85
8	10.44	-21.18
9	13.05	-23.17

La combinación en los factores que más afectan la señal S/R como se observa en Figura 5 son ordenamiento Nivel 3 (mixto), α en Nivel 3 (3 clientes) e iteraciones Nivel 1 (50,000), es decir, son los que más influyen en la variación del % Gap. De igual manera en la Figura 4, se muestra que el factor ordenamiento en su Nivel 3 (mixto), α en su Nivel 2 (0.9) e iteraciones en el Nivel 1 (50,000), son los que tienen más efecto sobre la media en el % Gap. Por lo tanto, el DOE Taguchi muestra que para obtener el % Gap menor se recomienda establecer el nivel de cada parámetro como se define en el Tabla 8.

Paso 8. Realizar una prueba para confirmar el resultado

Los niveles mostrados en el Tabla 8 son los que se especificaron en el algorítmo G-VRPTW.

Tabla 8. Niveles sugeridos por Taguchi

Factor	Nivel
Ordenamiento	mixto
α	0.90
Iteraciones	50,000

En el conjunto que contiene el 20 % de las instancias y cuyo resultados se muestran en la Figura 6.A, se observo lo siguiente:

- En las instancias de 25 se reduce el % Gap en 6.29 unidades y en 50 clientes se minimiza la

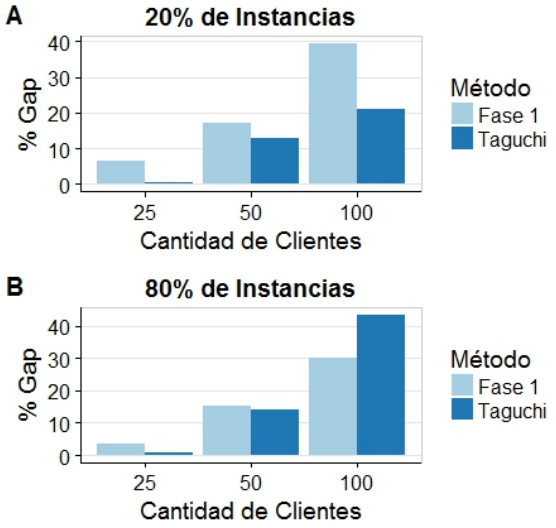

Fig. 6. Gráfica comparativa de los resultados obtenidos. A) en el 20 % de instancias y B) en el 80 % de ellas

distancia, ya que el % Gap obtenido disminuyó en 4.24 unidades al reportado en la Fase 1.

- En el comportamiento de las instancias de 100 clientes, el resultado muestra una disminución en 18.32 unidades, al compararlo con los valores obtenidos con calibración manual.

Al ejecutar el 80 % de las instancias, en la Figura 6.B, se observan los siguientes resultados:

- Las instancias de 25 se reduce 2.88 unidades y en 50 clientes se logra minimizar la distancia, ya que el % Gap obtenido disminuyó en 1.03 unidades que el reportado en la Fase 1.
- En el comportamiento de las instancias de 100 clientes, el resultado muestra un incremento del 13.48 unidades, al compararlo con los valores obtenidos con calibración manual.

Al analizar los resultados obtenidos ejecutando el algoritmo G-VRPTW con el vector P^* propuesto por el DOE Taguchi, mostrados en la Tabla 9, en donde el % Gap obtenido es producto de un promedio de 10 réplicas realizadas por cada una de las instancias. Se puede apreciar que: en las instancias de 25 clientes, se ha logrado un Gap menor al 10 %, destacando que en dos grupos de instancias se ha mejorado la solución reportada en la literatura; tal es el caso de las instancias RC1 y RC2, que de acuerdo a las características de las instancias el algoritmo G-VRPTW favorece aquellas que su distribución geográfica es mixta (RC), superando en todos los casos a los resultados obtenidos por la calibración empírica.

Tabla 9. Promedios del % Gap en el 80 % de las instancias

Resultados con Fase 1						
Clientes	***C1***	***C2***	***R1***	***R2***	***RC1***	***RC2***
25	1.40	10.00	2.5	6.10	0.10	6.10
50	**8.60**	23.10	**7.20**	27.90	**7.60**	27.90
100	**25.80**	**33.10**	**13.30**	36.90	31.90	**36.90**
Resultados utilizando Método Taguchi						
Clientes	***C1***	***C2***	***R1***	***R2***	***RC1***	***RC2***
25	**0.25**	**0.97**	**1.67**	**2.77**	**-0.65**	**-0.68**
50	10.45	**15.23**	10.70	**17.72**	9.35	**21.15**
100	74.22	71.08	20.69	**35.10**	**24.20**	37.74

Analizando los resultados en las instancias de 50 clientes se reflejan valores variables, en un rango entre 9.35 a 21.1 % de Gap, favoreciendo las instancias con ventanas amplias y de horizonte de programación largo con el vector de parámetros de Taguchi; en tanto los resultados obtenidos por calibración manual, en tres casos lo superan en promedio por 1.03 unidades. En los resultados reflejados en las instancias de 100 clientes se observan que la mayoría están alejados de la mejor solución conocida en la literatura revisada.

Para el grupo de instancias C2 los mejores resultados se presentan en la configuración 9 para esta clase de instancias, y la configuración 8 sugerida por Taguchi presentó, en el peor caso un Gap de 74.22 %, frente al 25.80 % obtenido por la configuración manual, lo que muestra un efecto negativo al considerar las medias, en un grupo de datos que no se ajusta totalmente a la normalidad Figura 3.

Después de ejecutar G-VRPTW con los niveles de los factores sugeridos al emplear el DOE

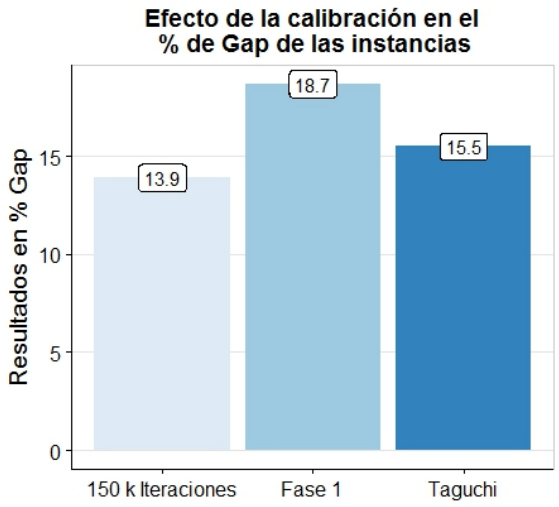

Fig. 7. Gráfica comparativa de los resultados finales obtenidos en las aproximaciones a la resolución del problema

Taguchi, se realizó un análisis de las soluciones obtenidas en el rango de 0 a 50,000 iteraciones, con el objetivo de determinar cuántas iteraciones fueron necesarias para obtener una mejor solución.

Se observó que el 41 % de las mejores soluciones se encuentran en el rango de 0 a 25,000 y el 51 % se generaron en el rango de 25,000 a 50,000. Ante esta tendencia se exploró el ir aumentando las iteraciones manteniendo los parámetros de ordenamiento y α en los niveles establecidos por el DOE Taguchi.

En la Tabla 10 se muestra que al ejecutar el algoritmo con 150,000 iteraciones (150k) en el conjunto de instancias de 25 clientes, se obtuvo un valor negativo, esto representa una solución con menor costo en la distancia que el mejor conocido en las instancias de Solomon [47].

En lo que respecta al costo de CPU al aumentar el número de iteraciones se presenta un comportamiento lineal, ya que al triplicar el número de iteraciones el tiempo presenta similar comportamiento.

Tabla 10. Promedio de % Gap por tipo de calibración

% inst.	Núm. clien.	Fase uno	Tagu-chi	CPU seg.	150 k	CPU seg.
20 %	25	6.69	0.40	6.21	-0.17	18.8
	50	17.14	12.90	22.82	11.02	69.2
	100	39.42	21.10	75.58	20.47	229.7
80 %	25	3.60	0.72	6.25	0.05	18.7
	50	15.13	14.10	21.26	10.68	63.7
	100	30.00	43.48	78.89	41.30	236.7

8.1. Comparación de resultados con otras investigaciones

En [16], se muestran resultados de 6 instancias de 25 clientes y 3 de 100 clientes, utilizan un algoritmo memético, además aplica una heurística evolutiva para establecer el cruzamiento y mutación entre subrutas, buscando obtener resultados más eficientes para resolver el VRPTW. En la Tabla 11 se compara las soluciónes encontradas (Dist.) en [16], además del tiempo computacional en segundos (t seg), teniendo un mejor comportamiento G-VRPTW.

Los resultados de las instancias de 25 y 50 clientes se compararon con los trabajos de [27] y de [23]. En la Tabla 12, se presenta un promedio de las instancias por grupo, el número de vehículos (V) y la distancia (Dist.). En el trabajo de [27] se resuelven las instancias de Solomon con métodos exactos basados en generación de columnas y en [23], se resuelve el VRPTW con la metaheurística de búsqueda Tabú.

Al comparar con G-VRPTW se puede observar que solo en el grupo de instancias RC1 de 25 clientes se logró mejorar los resultados de distancia presentados por [27] y con respecto a [23], en 5 grupos de instancias se obtuvieron resultados por abajo de los que ellos reportan.

Sin embargo, como en la función objetivo que se plantea en G-VRPTW no considera minimizar vehículos, en la Tabla 12 se observa en promedio un vehículo más que los que se reportan por [27] y [23].

Las instancias de 100 clientes se compararon con los trabajos de: Kontoravdis [31] y Chao-valitwongse [11], ambos construyen soluciones

ISSN 2007-9737

Tabla 11. Comparación de Resultados del G-VRPTW contra un algoritmo memético [16]

	[16]		G-VRPTW	
Instancia	Dist.	t(seg)	Dist.	t(seg)
C104-25 *vehículos*	190.13	42.2	188.44	18.3
R104-25 *vehículos*	470.25	47.8	417.95	21.47
RC108-25 *vehículos*	304.85	49.9	290.44	20.06
C204-25 *vehículos*	288.32	47.9	215.53	18.86
R208-25 *vehículos*	331.32	49.1	329.92	20.44
RC208-25 *vehículos*	288.52	52.9	269.52	5.68
C104-100 *vehículos*	1381.212	NR	1172.612	246.5
R104-100 *vehículos*	1414.114	NR	1232.914	283.66
RC108-100 *vehículos*	1851.312	NR	1420.715	88.52

NR = No resultados

Tabla 12. Comparación de los resultados del *G-VRPTW* de las instancias de 25 y 50 clientes contra otras investigaciones de la literatura

	[27]		[23]		G-VRPTW	
instancia	*V*	*dist.*	*V*	*Dist.*	*V*	*Dist.*
R1-25	5.0	463.36	4.75	470.17	5.8	471.63
R1-50	7.9	765.64	7.78	797.01	9.7	866.95
R2-25	2.7	382.14	1.27	430.77	3.2	393.12
R2-50	4.7	607.97	2	680.48	4.2	662.6
C1-25	3.0	190.58	3	191.08	4.0	191.17
C1-50	5.0	361.68	4.77	362.52	6.0	385.47
C2-25	1.8	214.45	1.13	246.36	3.0	215.34
C2-50	2.8	357.5	2	403.14	3.8	403.4
RC1-25	3.3	350.19	3.3	351.09	4.2	339.97
RC1-50	6.5	730.32	6.5	732.00	8	776.19
RC2-25	3.0	306.47	1.38	403.16	3.5	315.9
RC2-50	5.0	564.77	3.6	585.24	4.8	675.15

restringiendo el número de rutas e insertando clientes de acuerdo a la distancia máxima del cliente al depósito, y al cliente que requiera el servicio más temprano. En la búsqueda local elimina rutas sobrantes y aplica la heurística 2-OPT propuesta por [14], la cual consiste en que dada una solución inicial, selecciona 2 clientes de una ruta y los conecta a una diferente ruta, siempre y cuando mejore la solución y cumpla con las restricciones.

En [31] se utilizan todas las instancias de 100 clientes [46], las ejecuta 250 veces cada una y realiza 250 iteraciones, [11] utiliza una o dos instancias por cada grupo de 100 clientes, realiza 5 ejecuciones con 50 iteraciones. Ambos utilizan una lista de candidatos de 3 clientes.

Por su parte Repoussis en [43], utiliza un esquema de construcción de soluciones para GRASP, aplicando la inserción en paralelo, con una función codiciosa basada en penalizaciones. En la búsqueda local aplica el VNS (búsqueda variable de vecindario), propuesta por [36], que consiste en explorar vecindarios cada vez más distantes de la solución que tenga registrada hasta ese momento, moviéndose a una nueva si y sólo si es mejorada.

Después de obtener soluciones con la metaheurística GRASP Repoussis aplica la metaheuística de búsqueda Tabú [39], técnica basada en la inteligencia artificial, utilizada en la resolución de problemas combinatorios. Una de las principales características de esta metaheurística es que utiliza memoria flexible de búsqueda, la cual es utilizada para modificar, restringir y expandir sobre el entorno de vecindad.

En [35] se construye la ruta con el método GRASP de una manera voraz, se escogen las mejores soluciones para aplicar el algoritmo optimización por cúmulo de partículas [29] (por sus siglas en inglés, PSO), hace referencia a una metaheurÍstica que evoca el movimiento coordinado de los organismos vivos como una bandada de aves o un banco de peces.

En la Tabla 13 se reporta la solución obtenida por el conjunto de instancias, el tiempo computacional en segundos (CPU) y el número de vehículos (V); se muestran los resultados, que comparados con los obtenidos por el G-VRPTW se aprecia que el algoritmo no funciona bien para instancias de 100 clientes, aún así en las

instancias R1 y RC1 los resultados obtenidos en distancia están por arriba de [11].

Tabla 13. Comparación de los resultados del *G-VRPTW* contra investigaciones que utilizan GRASP en instancias de 100 clientes

Inst.	[31]	[11]	*[43]*	[35]	G-VRPTW
R1	1325.4	1603	1220.9	1192.1	1512.6
CPU	73.1	NR	NR	NR	181.39
V	12.6	18.5	NR	13.8	17.7
R2	1164.2	1268.5	974.4	821.3	1330.7
CPU	115.6	NR	NR	NR	158.15
V	3.1	4	NR	4.9	6.0
C1	827.3	885	828.38	833.3	1278.4
CPU	72.5	NR	NR	NR	151.62
V	10	10.5	NR	10	14.0
C2	589.6	598	589.86	666.2	972.5
CPU	127.5	NR	NR	NR	131.16
V	3	3	NR	3.4	5.8
RC1	1500.9	1721	1396.6	1354.6	1712.8
CPU	9	NR	NR	NR	177.7
V	12.6	15	NR	10.8	17.0
RC2	141.2		1160.9	1014.6	1546
CPU	13.51	NR	NR	NR	151.57
V	3.5			6	7.7

NR = No resultados

Finalmente, comparando los resultados con los mejores conocidos, reportados en [47] como se muestra en la Tabla 14, donde se reporta la solución obtenida (dist.) y el número de vehículos (V), que en las instancias R1, R2 y RC1 de 25 clientes, la distancia obtenida con el G-VRPTW está por abajo del reportado, al igual que las instancias R1 y RC2 de 50 clientes.

Una siguiente etapa sería minimizar los vehículos, que como ya se ha mencionado no se consideró en el presente trabajo y aún así se observa en las instancias R1-25, R2-25, R2-50, C2-25, RC1-25, RC2-25 y RC2-50, que la diferencia no es significativa, ya que el incremento es menor a uno. En el resto, en las instancias de 50 clientes, el incremento es en promedio de uno y en las de 100 clientes, el incremento ya es en promedio de cuatro vehículos.

9. Discusión

Los resultados mostrados representan una aproximación a los parámetros que se deben de considerar en el G-VRPTW, como se indica en [1] y se comprobó con la exploración adicional al modificar la cantidad de iteraciones. Aún con la anterior limitante se pudo observar que, el algoritmo presenta un alto rendimiento en las instancias de 25 clientes, como observamos en la Tabla 10, donde se indica que se logra mejorar el promedio de la mejor solución reportada en la literatura.

Tabla 14. Comparación de resultados del *G-VRPTW* contra mejores conocidos en la literatura [47]

	[47]		G-VRPTW	
Instancia	*V*	*dist.*	*V*	*dist.*
R1-25	4.91	481.65	5.4	461.40
R1-50	7.75	796.86	8.75	89.51
R1-100	11.92	1210.30	17.7	1512.6
R2-25	2.72	384.67	2.8	375.55
R2-50	4.11	654.10	4.2	759.67
R2-100	2.7	951.03	6.0	1330.7
C1-25	3.0	190.50	4.0	191.45
C1-50	5.0	361.60	6.3	386.90
C1-100	10.0	827.10	14.0	1278.4
C2-25	2	214.45	2.8	216.52
C2-50	2.7	337.50	4.0	401.45
C2-100	3.0	587.9	5.8	972.5
RC1-25	3.3	359.25	4.1	329.98
RC1-50	6.5	731.32	7.5	731.78
RC1-100	11.4	1396.45	17.0	1712.8
RC2-25	2.8	319.27	3.3	307.21
RC2-50	4.4	608.75	4.6	693.78
RC2-100	3.3	1129.23	7.7	1546.0

En las instancias de 50 clientes, en promedio se tiene un Gap del 10 % respecto a la mejor solución y en las instancias de 100 clientes, es indiscutible la necesidad de explorar la superficie con base en los resultados obtenidos en el presente trabajo para reducir el diferencial existente con la mejor solución conocida.

Las recomendaciones que se concluyen de ésta investigación es que al utilizar la metaheurística G-VRPTW, el vector P^* debe considerar un α con 0.90, ya que de acuerdo a los resultados obtenidos este parámetro nos permite mejorar los

resultados al compararlo con la selección de 3 clientes que se recomienda en la literatura, para problemas similares. Es recomendable agregar el objetivo de minimizar el número de vehículos, es decir, resolver el G-VRPTW con un problema multiobjetivo.

Finalmente como se aprecia en la Figura 7, los resultados obtenidos muestran que al establecer el vector P^* mediante un método de calibración se mejora la calidad de las soluciones del algoritmo G-VRPTW, tal como se aprecia al comparar la Fase 1, con un Gap de 18.7 %, con el obtenido al utilizar el DOE Taguchi descendiendo al 16.6 %. Al aumentar el número de iteraciones y mantener el vector P^* Taguchi, el Gap disminuye a un 13.9 % en promedio de la mejor solución reportada en la literatura.

10. Conclusiones y trabajo futuro

Una vez utilizado el DOE Taguchi, se recomienda efectuar un análisis de superficie de respuesta, es decir utilizar técnicas matemáticas y estadísticas para ajustar la variable de respuesta (distancia) a un polinomio de primer o segundo grado e intentar aproximar a un punto estacionario que minimiza la respuesta [38] , esto con el fin de establecer los parámetros que mejoren los resultados del algorítmo G-VRPTW, sobre todo para las instancias de 100 clientes, como se realiza en la propuesta de la *herramienta de Calibra*, que utiliza primero la metodología de Taguchi y después aplica una búsqueda local [1].

En lo que respecta al trabajo futuro, se deben incluir otros indicadores de desempeño diferentes a la calidad de la solución como por ejemplo el número de vehículos, el esfuerzo computacional y la robustez de los parámetros.

Considerando las nuevas tendencias en las investigaciones relacionadas con el problema de rutas de vehículos y la logística verde [45], se propone considerar vehículos eléctricos en la aplicación del G-VRPTW.

Agradecimientos

Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por las becas otorgadas a Danisa Romero, Víctor Valenzuela, Gener Avilés para realizar sus estudios de maestría. Danisa Romero y Víctor Valenzuela agradecen al Tecnológico Nacional de México en Agua Prieta, Sonora; el apoyo brindado en los estudios de posgrado.

Referencias

1. **Adenso-Diaz, B. & Laguna, M. (2006).** Fine-tuning of algorithms using fractional experimental designs and local search. *Operations research*, Vol. 54, No. 1, pp. 99–114.
2. **Al-Ghazzawi, A., Ali, E., Nouh, A., & Zafiriou, E. (2001).** On-line tuning strategy for model predictive controllers. *Journal of Process Control*, Vol. 11, No. 3, pp. 265–284.
3. **Alssager, M. & Othman, Z. A. (2016).** Taguchi-based parameter setting of cuckoo search algorithm for capacitated vehicle routing problem. In *Advances in Machine Learning and Signal Processing*. Springer, pp. 71–79.
4. **Amini, M. M. & Barr, R. S. (1993).** Network reoptimization algorithms: A statistically designed comparison. *ORSA Journal on Computing*, Vol. 5, No. 4, pp. 395–409.
5. **Amini, M. M. & Racer, M. (1994).** A rigorous computational comparison of alternative solution methods for the generalized assignment problem. *Management Science*, Vol. 40, No. 7, pp. 868–890.
6. **Anderson, T. W. & Darling, D. A. (1952).** Asymptotic theory of certain"goodness of fitçriteria based on stochastic processes. *The annals of mathematical statistics*, pp. 193–212.
7. **Barr, R. S., Golden, B. L., Kelly, J. P., Resende, M. G., & Stewart, W. R. (1995).** Designing and reporting on computational experiments with heuristic methods. *Journal of heuristics*, Vol. 1, No. 1, pp. 9–32.
8. **Birattari, M. & Kacprzyk, J. (2009).** *Tuning metaheuristics: a machine learning perspective*, volume 197. Springer.
9. **Bovet, D. M. M. & Bob, W. (2000).** Camino sin obstáculos. *GestiónVol. 5, no. 1 (2000 ene.), p. 70-74*.

10. **Bräysy, O. & Gendreau, M. (2005).** Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation science*, Vol. 39, No. 1, pp. 104–118.
11. **Chaovalitwongse, W., Kim, D., & Pardalos, P. M. (2003).** Grasp with a new local search scheme for vehicle routing problems with time windows. *Journal of Combinatorial Optimization*, Vol. 7, No. 2, pp. 179–207.
12. **Cosıo-León, M., Martınez-Vargas, A., & Gutierrez, E.**, **.** An experimental study of parameter selection in particle swarm optimization using an automated methodology. *Advances in Artificial Intelligence*, pp. 9.
13. **Coy, S. P., Golden, B. L., Runger, G. C., & Wasil, E. A. (2001).** Using experimental design to find effective parameter settings for heuristics. *Journal of Heuristics*, Vol. 7, No. 1, pp. 77–97.
14. **Croes, G. A. (1958).** A method for solving traveling-salesman problems. *Operations research*, Vol. 6, No. 6, pp. 791–812.
15. **Crowder, H. P., Dembo, R. S., & Mulvey, J. M. (1978).** Reporting computational experiments in mathematical programming. *Mathematical Programming*, Vol. 15, No. 1, pp. 316–329.
16. **Cruz Chávez, M. A. & Díaz Parra, O. (2010).** Evolutionary algorithm for the vehicles routing problem with time windows based on a constraint satisfaction technique. *Computación y Sistemas*, Vol. 13, No. 3, pp. 257–272.
17. **Dantzig, G. B. & Ramser, J. H. (1959).** The truck dispatching problem. *Management science*, Vol. 6, No. 1, pp. 80–91.
18. **Dobslaw, F. (2010).** Recent development in automatic parameter tuning for metaheuristics. *Proceedings of the 19th Annual Conference of Doctoral Students-WDS*, volume 2010, pp. 54–63.
19. **Feo, T. A. & Resende, M. G. (1989).** A probabilistic heuristic for a computationally difficult set covering problem. *Operations research letters*, Vol. 8, No. 2, pp. 67–71.
20. **Feo, T. A. & Resende, M. G. (1995).** Greedy randomized adaptive search procedures. *Journal of global optimization*, Vol. 6, No. 2, pp. 109–133.
21. **Fukunaga, A. S. (2008).** Automated discovery of local search heuristics for satisfiability testing. *Evolutionary computation*, Vol. 16, No. 1, pp. 31–61.
22. **Greenberg, H. J. (1990).** Computational testing: Why, how and how much. *ORSA Journal on Computing*, Vol. 2, No. 1, pp. 94–97.
23. **Hedar, A.-R. & Bakr, M. A. (2014).** Three strategies tabu search for vehicle routing problem with time windows. *Computer Science and Information Technology*, Vol. 2, No. 2, pp. 108–119.
24. **Hoaglin, D. C. & Andrews, D. F. (1975).** The reporting of computation-based results in statistics. *The American Statistician*, Vol. 29, No. 3, pp. 122–126.
25. **Hutter, F., Hoos, H. H., Leyton-Brown, K., & Stützle, T. (2009).** Paramils: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, Vol. 36, No. 1, pp. 267–306.
26. **Hutter, F., Hoos, H. H., & Stützle, T. (2007).** Automatic algorithm configuration based on local search. *AAAI*, volume 7, pp. 1152–1157.
27. **Kallehauge, B., Larsen, J., & Madsen, O. B. (2001).** Imm.
28. **Kelleher, J. D., Mac Namee, B., & D'Arcy, A. (2015).** *Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies*. MIT Press.
29. **Kennedy, J. (2011).** Particle swarm optimization. In *Encyclopedia of machine learning*. Springer, pp. 760–766.
30. **Kiremire, A. R. (2011).** The application of the pareto principle in software engineering. *Consulted January*, Vol. 13, pp. 2016.
31. **Kontoravdis, G. & Bard, J. F. (1995).** A grasp for the vehicle routing problem with time windows. *ORSA journal on Computing*, Vol. 7, No. 1, pp. 10–23.
32. **Laporte, G., Gendreau, M., Potvin, J.-Y., & Semet, F. (2000).** Classical and modern heuristics for the vehicle routing problem. *International transactions in operational research*, Vol. 7, No. 4-5, pp. 285–300.
33. **Limon-Romero, J., Tlapa, D., Baez-Lopez, Y., Maldonado-Macias, A., & Rivera-Cadavid, L. (2016).** Application of the taguchi method to improve a medical device cutting process. *The International Journal of Advanced Manufacturing Technology*, Vol. 87, No. 9-12, pp. 3569–3577.
34. **López-Ibánez, M., Dubois-Lacoste, J., Stützle, T., & Birattari, M. (2011).** The irace package, iterated race for automatic algorithm configuration. Technical report, Citeseer.

35. **Mao, Y. & Deng, Y. (2010).** Solving vehicle routing problem with time windows with hybrid evolutionary algorithm. *Intelligent Systems (GCIS), 2010 Second WRI Global Congress on*, volume 1, IEEE, pp. 335–339.
36. **Mladenović, N. & Hansen, P. (1997).** Variable neighborhood search. *Computers & operations research*, Vol. 24, No. 11, pp. 1097–1100.
37. **Montgomery, D. (1997).** *Montgomery Design and Analysis of Experiments*. John Wiley.
38. **Montgomery, D. C. (2010).** *Diseños y análisis de experimentos*.
39. **Piqueras, V. Y. (2008).** *Optimización heurística económica aplicada a las redes de transporte del tipo VRPTW.* Ph.D. thesis.
40. **Pulido, H. G., De la Vara Salazar, R., González, P. G., Martínez, C. T., & Pérez, M. d. C. T. (2004).** *Análisis y diseño de experimentos*. McGraw-Hill.
41. **Pullen, H. & Webb, M. (1967).** A computer application to a transport scheduling problem. *The Computer Journal*, Vol. 10, No. 1, pp. 10–13.
42. **Rajkumar, M., Asokan, P., Anilkumar, N., & Page, T. (2011).** A grasp algorithm for flexible job-shop scheduling problem with limited resource constraints. *International Journal of Production Research*, Vol. 49, No. 8, pp. 2409–2423.
43. **Repoussis, P. P., Paraskevopoulos, D. C., Tarantilis, C. D., & Ioannou, G. (2006).** A reactive greedy randomized variable neighborhood tabu search for the vehicle routing problem with time windows. *International Workshop on Hybrid Metaheuristics*, Springer, pp. 124–138.
44. **Resende, M. G. & Ribeiro, C. C. (2010).** Greedy randomized adaptive search procedures: Advances, hybridizations, and applications. In *Handbook of metaheuristics*. Springer, pp. 283–319.
45. **Sbihi, A. & Eglese, R. W. (2010).** Combinatorial optimization and green logistics. *Annals of Operations Research*, Vol. 175, No. 1, pp. 159–175.
46. **Solomon, M. M. (1987).** Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, Vol. 35, No. 2, pp. 254–265.
47. **Solomon, M. M. (2005).** Vrptw benchmark problem. http://web.cba.neu.edu/ msolomon/problems.htm.
48. **Toth, P. & Vigo, D. (2014).** *Vehicle routing: problems, methods, and applications*. SIAM.
49. **Van Breedam, A. (1995).** Improvement heuristics for the vehicle routing problem based on simulated annealing. *European Journal of Operational Research*, Vol. 86, No. 3, pp. 480–490.
50. **Wolpert, D. H. & Macready, W. G. (1997).** No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, Vol. 1, No. 1, pp. 67–82.
51. **Xu, J., Chiu, S. Y., & Glover, F. (1998).** Fine-tuning a tabu search algorithm with statistical tests. *International Transactions in Operational Research*, Vol. 5, No. 3, pp. 233–244.
52. **Yaghini, M. & Kazemzadeh, M. R. A. (2012).** Dimma: A design and implementation. *Modeling, Analysis, and Applications in Metaheuristic Computing: Advancements and Trends: Advancements and Trends*, pp. 90.
53. **Zemel, E. (1981).** Measuring the quality of approximate solutions to zero-one programming problems. *Mathematics of operations research*, Vol. 6, No. 3, pp. 319–332.

Article received on 28/03/2017; accepted on 15/08/2017.
Corresponding author is Alma Danisa Romero-Ocaño.