IMPLEMENTACIÓN DE UN ALGORITMO BASADO EN LA TÉCNICA HÍBRIDA METAHEURÍSTICA HAS-QAP EN EL PROBLEMA DE ASIGNACIÓN CUADRÁTICA Y APLICACIÓN DE TÉCNICAS ESTADÍSTICAS MULTIVARIADAS PARA LA IDENTIFICACIÓN DE INTERRELACIONES ENTRE ALGUNOS DE LOS MÉTODOS DISPONIBLES PARA RESOLVERLO

RAQUEL SALAS RIVERA

UNIVERSIDAD DE LOS ANDES
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INDUSTRIAL
MAESTRÍA EN INGENIERÍA INDUSTRIAL
BOGOTÁ D.C.
2006

IMPLEMENTACIÓN DE UN ALGORITMO BASADO EN LA TÉCNICA HÍBRIDA METAHEURÍSTICA HAS-QAP EN EL PROBLEMA DE ASIGNACIÓN CUADRÁTICA Y APLICACIÓN DE TÉCNICAS ESTADÍSTICAS MULTIVARIADAS PARA LA IDENTIFICACIÓN DE INTERRELACIONES ENTRE ALGUNOS DE LOS MÉTODOS DISPONIBLES PARA RESOLVERLO

Trabajo de grado presentado para optar al título
de Magíster en Ingeniería Industrial

RAQUEL SALAS RIVERA

Asesor

JOSÉ FIDEL TORRES DELGADO

UNIVERSIDAD DE LOS ANDES
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INDUSTRIAL
MAESTRÍA EN INGENIERÍA INDUSTRIAL
BOGOTÁ D.C.
2006

RECONOCIMIENTOS

Agradecimientos al profesor José Fidel Torres por su colaboración durante todo el proceso de este trabajo y al profesor Robinson Rico por su interés y tiempo dedicado.

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

Capítulo I

INTRODUCCIÓN

La importancia de la logística en la drástica mejora de la competitividad ha llevado a una investigación tecnológica aplicada que permite convertir información en tiempo real, en productos y/o servicios al consumidor en cualquier parte del mundo a precios competitivos y en tiempo record.

Dentro del grupo de nuevas tecnologías aplicadas, destinadas a la mejora en la eficiencia de la logística se encuentran las técnicas Metaheurísticas, basadas en algoritmos bio-inspirados que proponen una aproximación diferente a los modelos clásicos, para el diseño y la optimización de los sistemas.

Los algoritmos bio-inspirados como su nombre lo indica se inspiran en los sistemas naturales (biología y evolución) y tienen como características representativas: ser auto-adaptativos, ser-auto-didactas (capaces de aprender: tener memoria) y ser auto-organizados (auto-controlados). Deben su utilidad y acogida a la capacidad que tienen de resolver problemas complejos, como lo son en su mayoría los problemas de logística, y que son de gran dificultad para los métodos computacionales tradicionales. Por tal razón, se emplea con éxito en áreas tales como: optimización combinatoria, aprendizaje de máquinas, diseño de ingeniería entre otras.

Entre los algoritmos bio-inspirados se encuentran los sistemas basados en agentes, como lo son las técnicas *Algoritmos Hormigas* y *Optimización por Colonia de Hormiga* (ACO), en las cuales se basa este proyecto. Estas técnicas meta-heurísticas constituyen un enfoque alternativo a la solución de problemas de optimización combinatoria, tal como el problema NP-Hard QAP de asignación cuadrática el cual

apunta entre otras aplicaciones, a la reducción de las necesidades de almacenamiento, con la consiguiente reducción de costos y mejora en la rotación y el flujo de productos.

La idea central de ellas es la de un número de agentes artificiales simples (análogos a las hormigas) implementados como procesadores en paralelo cuyo objetivo es construir buenas soluciones a problemas de optimización combinatoria complejo gracias a la utilización de comunicaciones de bajo nivel (indirectas). Las hormigas reales cooperan en su búsqueda de alimento depositando en el suelo trazas de una sustancia química llamada feromona. Una colonia de hormigas artificiales simula este comportamiento mediante la utilización de una memoria común que corresponde a la feromona depositada por las hormigas reales.

Las soluciones se obtienen entonces, mediante un procedimiento jalonado por una combinación de la feromona artificial, los datos propios del problema y una función heurística (basada en leyes probabilísticas) empleada para evaluar los sucesivos pasos de la construcción.

El problema de asignación cuadrática (QAP) de orden *n* es uno de los problemas de optimización combinatoria compleja que ha sido resuelto empleando algoritmos de la meta heurística ACO y que hasta al momento han resultado ser tan exitosos, como lo son para encontrar buenas soluciones en el problema del agente viajero (TSP).

El modelo originario de ACO para resolver el problema QAP fue planteado por Maniezzo, Colorni y Dorigo en "*Ant System Applied to the Quadratic Assignment Problem*" y es reconocido genéricamente como la técnica *Sistema Hormigas* (AS) que aplicada al problema QAP toma la sigla AS-QAP. Este tradicional sistema emplea la información dejada por las huellas de feromona para construir las

soluciones completas al problema QAP. A partir de AS nuevos algoritmos han sido propuestos reteniendo algunas de las características de la inspiración biológica original. Sin embargo, cada vez son menos los inspirados en la analogía biológica y muchos más los motivados por la necesidad de volver más competitivos y eficientes la metaheurística ACO a través de mejoras basadas en el estado del arte de los algoritmos.

Dentro de los algoritmos recientes se encuentra uno muy interesante conocido como *Sistema de Hormigas Híbrido* (HAS) cuya aplicación a problemas QAP fue propuesto en el trabajo de Dorigo y Gambardella en "*Ant Colonies for the Quadratic Assignment Problem*" [1997], tomando el nombre de HAS-QAP. Este algoritmo, aunque hace uso en alguna medida de los aspectos de la analogía biológica, no puede ser considerado una instancia de la metaheurística ACO. A diferencia de los algoritmos de construcción que caracterizan a los ACO, en los HAS-QAP la huella de feromona no es usada para guiar la construcción de la solución, por el contrario, se emplea para guiar modificaciones en soluciones completas bajo el estilo de búsqueda local. Estos algoritmos pertenecen a una metaheurística más global que encierra todos los algoritmos basados en hormigas (aún la metaheurística ACO) conocida con el nombre de *Algoritmos Hormigas*.

Dorigo y Gambardella plantean a HAS-QAP como un algoritmo híbrido que resulta de la combinación de AS-QAP con la heurística de búsqueda local como técnica de mejoramiento. Resultados experimentales han demostrado que el modelo híbrido se desenvuelve mejor en problemas irregulares y estructurados, como son los problemas complejos de la vida real, debido a su capacidad para encontrar la estructura de las buenas soluciones en un corto tiempo computacional.

Capítulo II

DESCRIPCIÓN DEL PROBLEMA DE ASIGNACIÓN CUADRÁTICA QAP

El problema de asignación cuadrática (QAP) es un problema de gran importancia tanto en teoría como en la práctica que ha sido estudiado por numerosos investigadores en el campo de la optimización combinatoria durante la segunda mitad del siglo veinte. Dentro de los problemas reales que pueden ser formulados como QAP se encuentran: localización de instalaciones, problemas de cableado en electrónica, diseño de diseño y problemas de secuenciación de trabajos. El problema QAP es considerado un problema de optimización combinatoria (COP) declarado por primera vez por Koopmans y Beckman [1957] como la búsqueda de la mejor asignación de n actividades a n localidades (o viceversa). La demostración del problema QAP como problema NP-hard[1] fue hecha por Shani y Gonzalez [1976] y es considerado uno de los más duros dentro de esta clase (NP-hard) ya que mientras pocos problemas pueden ser resueltos exactamente para instancias relativamente largas como el problema del agente viajero (TSP), instancias QAP con $n \geq 20$ no pueden ser resueltas óptimamente. De esta forma la aproximación exacta esta limitada directamente por la complejidad del problema medida en función de n. Las instancias pequeñas QAP pueden ser resueltas óptimamente por algoritmos exactos que están limitados a técnicas como: ramificación y acotación y programación dinámica. La solución de instancias grandes en cambio, debido al pobre rendimiento de los algoritmos exactos (determinado en requerimiento excesivo de tiempo computacional) requiere de algoritmos aproximados conocidos como heurísticas, que reunidas en esquemas generales complejos toman el nombre de metaheuríticas.

[1] No existe ningún algoritmo que lo resuelva en tiempo polinomial asumiendo que $P \neq NP$ (Garey y Johnson [1979]). El término hard es capturado por la teoría de la *Complejidad Computacional*.

Aunque no aseguran encontrar la solución óptima proporcionan “buenos” resultados en un tiempo computacional aceptable.

2.1. FORMULACIÓN DEL PROBLEMA DE ASIGNACIÓN CUAUADRÁTICA QAP

El problema QAP de orden n puede ser descrito como la asignación de n actividades a n localidades (o viceversa) con distancias dadas entre localidades, y flujos dados entre actividades. El sistema de medida de las distancias puede estar basado en métricas como la Euclidiana, la de Manhattan entre otras; los flujos representan el flujo de material, personas, objetos, etc. de una actividad a otra. El objetivo es encontrar la asignación de actividades a localidades que minimice la suma de los productos entre flujos y distancias.

Matemáticamente el problema es definido por dos matrices de dimensión $n \times n$. La figura 1 muestra una posible representación gráfica de ambas.

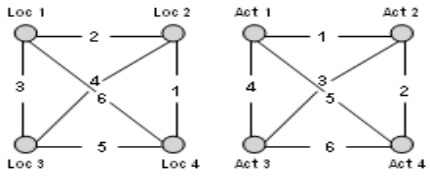


Figura 1. Representación de la matriz de distancias (izquierda) y de la matriz de flujos (derecha)

- *Matriz de distancias*: $A = \{a_{ij}\}$, a_{ij} representa la distancia entre la localidad i y la localidad j.
- *Matriz de flujos*: $B = \{b_{rs}\}$, b_{rs} representa el flujo entre la actividad r y la actividad s.

De acuerdo a la descripción anterior el problema QAP puede ser formulado de acuerdo a su orientación al algoritmo (la representación es definida con el fin de adaptar el modelo al algoritmo que será usado para resolverlo), o a su formulación como problema de optimización combinatoria.

2.1.1. Formulación del problema QAP orientada al algoritmo

Bajo esta formulación el problema QAP consiste en la búsqueda de una permutación $\pi \in \Pi$ de n elementos correspondiente a una posible solución del problema que minimice $f(\pi) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} b_{\pi(i)\pi(j)}$, donde Π representa el conjunto de todas las posibles permutaciones (soluciones) y $f(\pi)$ el valor de la función objetivo asociado a π. En una permutación $\pi : i \leftarrow \pi(i)$ puede ser interpretado como la asignación particular de la actividad $r = \pi(i)$ a la localidad $i(i = 1,2,...,n)$, así mismo $\pi : j \leftarrow \pi(j)$ como la asignación particular de la actividad $s = \pi(j)$ a la localidad $j(j = 1,2,...,n)$ en el caso en que se asignen actividades a localidades[2]. De esta forma el producto $a_{ij} b_{\pi(i)\pi(j)}$ describe la contribución del costo de la asignación simultánea de la actividad $\pi(i)$ a la localidad i y de la actividad $\pi(j)$ a la localidad j.

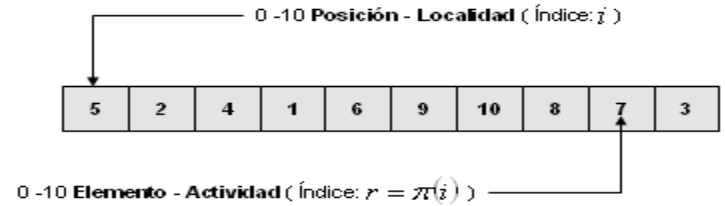


Figura 2. Ilustración de un vector-cromosoma para $n = 10$

[2] La asignación también puede darse de localidades a actividades donde el índice i corresponde a la actividad y el índice $r = \pi(i)$ a la localidad. En éste trabajo se utiliza de ahora en adelante la asignación de actividades a localidades.

En la mayoría de los problemas de optimización combinatoria la representación de la solución utilizada por los métodos heurísticos y metaheurísticos es a través de una permutación π modelada gráficamente por un vector-cromosoma como el que se observa en la figura 2.

2.1.2. Formulación del problema QAP como un problema de optimización entera

El problema QAP puede ser reformulado como un problema de optimización entera[3] con una función objetivo cuadrática. Resolver el problema significa hallar una matriz de permutaciones $X = \{x_{ij}\}$ de dimensión $n \times n$ donde x_{ij} es una variable binaria que toma el valor de 1 si la actividad j es asignada a la localidad i y 0 de lo contrario. El problema puede ser formulado de la siguiente forma:

Conjunto de localidades

I : Conjunto de todas las localidades (Índice: i)

Conjunto de actividades

J : Conjunto de todas las actividades (Índice: j)

$$\text{Minimizar} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{l=1}^{n} \sum_{k=1}^{n} a_{ij} b_{kl} x_{ik} x_{jl} \qquad (1)$$

Sujeto a:

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall_j \in J \qquad (2)$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall_i \in I \qquad (3)$$

$$x_{ij} \in \{0,1\} \qquad \forall_{i,j} \in (I, J) \qquad (4)$$

[3] Casi siempre un problema de optimización combinatoria puede ser formulado como uno de optimización entera (Dumitresco y Stützle [2003]).

La función objetivo (1) representa el propósito de encontrar la matriz de permutaciones X que minimice las sumatorias de los productos entre flujos y distancias de la asignación simultánea de la actividad j a la localidad i y de la actividad l a la localidad k. El conjunto de restricciones (2) aseguran que toda actividad sea asignada a una única localidad. Las restricciones (3) aseguran que a cada localidad se le sea asignada una única actividad. En (4) se declaran los posibles valores que puede tomar la variable como variable binaria.

2.2. INSTANCIAS DEL PROBLEMA QAP

El desempeño del método empleado depende fuertemente del tipo de problema a ser resuelto. De ahí que un método excelente para un determinado tipo de problema sea pésimo para otro. Por esta razón, son muy pocos los autores que comparan sus métodos con otros en muchos tipos diferentes de problemas (Taillard [1995]). La clasificación de las instancias fue motivada con el fin de agrupar de acuerdo a características comunes una muestra representativa de problemas proveniente de la literatura. La gran mayoría de ellos pueden ser encontrados en la librería QAPLIB de Burkard et al [1991]. Taillard [1995] propuso la clasificación de las instancias de QAPLIB dentro de cuatro clases:

(a) Instancias no estructuradas y generadas aleatoriamente: Instancias con matrices de flujo y distancia generadas aleatoriamente de acuerdo a una distribución uniforme. Dentro de ellas se encuentran las instancias **taixxa**, donde **xx** es el tamaño del problema considerado. Comúnmente los límites dentro de los cuales son generadas son 0 y 99. Son consideradas unas de las más difíciles de resolver óptimamente. Y dentro de la literatura han sido ampliamente usados por ser casi las únicas que no han sido resueltas pseudos-óptimamente.

(b) Instancias no estructuradas con distancias Manhattan: Las instancias consideradas dentro de esta clase se generan de la siguiente forma: un rectángulo constituido de $n_1 \times n_2$ cuadrados de tamaño unitario. Para el cálculo de la matriz de distancia una localidad corresponde a uno de los cuadrados y la distancia entre dos cuadrados es la distancia dada por la métrica de Maniatan medida entre ellos. La matriz de flujos es generada aleatoriamente, pero no necesariamente con distribución uniforme.

La característica de este tipo de problemas es que son simétricos y por tanto pueden tener entre cuatro $(n_1 \neq n_2)$ y ocho $(n_1 = n_2)$ diferentes soluciones óptimas. Entre ellos se encuentran los Nugent, Vollmann y Ruml [1968] (**nugxx**), Skorin-Kapov [1990] (**skoxx**) y Wilhelm y Ward [1987].

(c) Instancias de la vida real: En esta clase se han agrupado los problemas que han sido el resultado de aplicaciones prácticas en la vida real. La lista de ellas es siempre extensa, pero las más utilizadas en la literatura son: instancia de Steinberg (**ste36x**)-donde la **x** corresponde al problema específico representado por una letra del abecedario-, instancia de Elshafei (**els19**), el problema del diseño de un hospital (**kra30x**), instancias asociadas al diseño de teclados de computadoras (**bur26x**) y un nuevo tipo de instancias propuestas en [35]. Las instancias de la vida real son caracterizadas por dos aspectos:

- La matriz de flujo tiene una gran cantidad de componentes con valor de cero y el resto de ellos no tienen una distribución uniforme claramente definida.

- Este aspecto comprende la estructura del óptimo local. Este tipo de problemas tienen "alguna" estructura reflejada por los componentes de la matriz de flujo que puede ser encontrada examinando el óptimo local. Esto significa que las

permutaciones correspondientes al óptimo local privilegian ciertas posiciones (localidades) con determinados elementos (actividades). La *entropía*[4] de un grupo de permutaciones definido por Fleurent y Ferland [1994] mide este privilegio de la siguiente forma: sea m el número soluciones consideradas y n_{ij} el número de veces que la actividad j es asignada a la localidad i en las m soluciones. Se definen los siguientes valores

$$v_{ij} = \begin{cases} 0 & \text{Si } n_{ij} = 0 \\ -\frac{n_{ij}}{m} \log \frac{n_{ij}}{m} & \end{cases} \qquad (1)$$

La entropía de las m soluciones esta dada por:

$$E = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} v_{ij}}{n \log n} \qquad (2)$$

De acuerdo a (2) un grupo de soluciones (permutaciones) idénticas tiene una entropía de $E = 0$ y un grupo de soluciones uniformemente distribuidas en un conjunto de m permutaciones de n elementos tiene una entropía que tiende a 1 cuando m tiende a infinito.

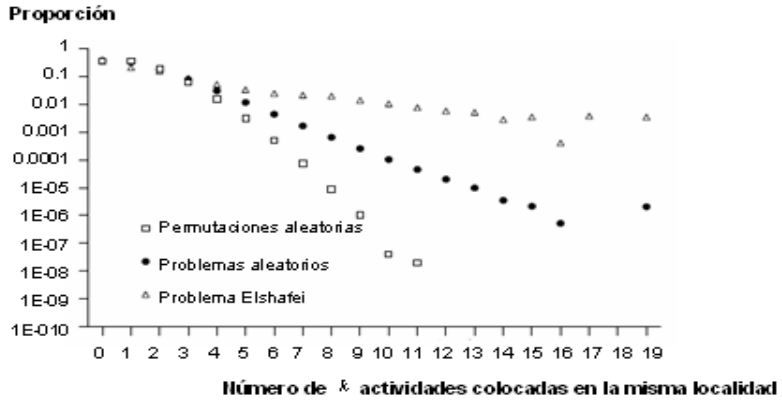


Figura 3. Proporción de actividades asignadas a determinadas localidades

[4] Magnitud dinámica que proporciona una medida del grado de desorden de un sistema [Salvat].

Con base a este parámetro (entropía) Fleurent y Ferland [1994] empleando la instancia **els19** demostraron que las instancias de la vida real tienen más estructura que las instancias generadas unifórmente. La ilustración intuitiva de ello se observa en la figura 3 como una función de k. Se calcula la proporción de soluciones que tienen k actividades con diferentes valores de j $(j = 1,2,...,n)$ asignadas a la misma localidad. La proporción decrece muy rápido cuando las soluciones provienen de una población distribuida uniformemente; decrece más lento cuando las soluciones provienen de óptimos locales asociados a problemas generados aleatoriamente y por último se observa que una relativamente alta proporción del óptimo local del problema Elshafei (instancia **els19**) difiere por pocas actividades.

Para llegar a la demostración citada primero determinaron que uno de cada cincuenta óptimos locales eran también óptimos globales. Luego se percataron que con las primeras soluciones iniciales obtenidas, tres de las $n = 19$ actividades eran asignadas a las mismas localidades a las que estaban asignadas en los óptimos globales y el resto de ellas estaban distribuidas aleatoriamente en la permutación. En últimas observaron que cerca de la mitad de los óptimos locales eran al mismo tiempo óptimos globales. Lo anterior llevó a concluir que cuando se tiene conocimiento previo sobre la asignación de unas pocas actividades (localidades a donde deben ser ubicadas) el problema se vuelve muy fácil. Y por ende entre más rápido se encuentre la estructura de estas pocas asignaciones en problemas de este tipo más eficiente será considerado el método empleado para resolverlos.

La explicación encontrada en los problemas de la vida real es que casi siempre la situación dada presenta una alta interacción entre una parte de sus actividades (ítenes-unidades, etc.). De lo anterior se deduce que en las buenas soluciones las actividades con una alta interacción con otras deberían ser asignadas a localidades interconectadas por distancias relativamente pequeñas (Stützle [1997]).

(d) Instancias aleatorias no uniformes: Debido a que la mayoría de las instancias de la vida real eran de tamaño pequeño no se podía comparar métodos en una gama variada de problemas de diferentes tamaños. Por otra parte los problemas generados aleatoriamente que eran usados en la literatura no se consideraban interesantes. Estas dos situaciones motivaron a la construcción de problemas que en lo posible tuvieran las mismas características de las instancias de la vida real. Tales problemas son generados aleatoriamente mas no unifórmente.

2.3. IDENTIFICACIÓN DEL TIPO DE INSTANCIA QAP

La diferenciación de las instancias QAP dentro de cada una de las clases mencionadas se basa en la expresión de la dificultad propia del problema para ser resulto, a través de medidas estadísticas fáciles de calcular. La más conocida es la *dominancia del flujo* ($fd(B)$) definida como el coeficiente de variación de los componentes de la matriz de flujo multiplicada por 100.

$$fd(B) = 100 \cdot \frac{\sigma}{\mu}, \text{ donde} \tag{5}$$

$$\mu = \frac{\sum_{r=1}^{n}\sum_{s=1}^{n} b_{rs}}{n^2} \tag{6}$$

$$\sigma = \sqrt{\frac{\sum_{r=1}^{n}\sum_{s=1}^{n} (b_{rs} - \mu)}{n^2 - 1}} \tag{7}$$

De acuerdo a (5) un valor alto de $fd(B)$ indica que una gran parte de la matriz de flujo esta constituida por unas pocas actividades. De esa forma las instancias consideradas dentro de la clase (a) (aquellas generadas aleatoriamente con

distribución uniforme) tendrán valores de $fd(B)$ relativamente bajos, mientras que las clasificadas como (c) (problemas de la vida real) tendrán en general, valores de $fd(B)$ considerablemente altos.

La desventaja de esta medida era que sólo capturaba la estructura de la matriz de flujos y dejaba a un lado la matriz de distancias. Por esa razón, se definió análogamente la estadística *dominancia de la distancia* ($dd(A)$). En el trabajo de Stützle [1997] se calcularon los valores de dominancia para ambas matrices y los resultados estos mostraron que las instancias clasificadas dentro de las clases (c) y (d) tienen valores de dominancia considerablemente altos en al menos una de las matrices. La tabla 1 resume los resultados de $fd(B)$ y $dd(A)$ correspondientes a las instancias seleccionadas para cada clase.

Por último, la medida de *sparsity* (sp) de una matriz fue definida por Dorigo y Stützle [2001] después de observar la cantidad de entradas con valor cero que tenían en general las matrices de los problemas de la vida real. De ahí se consideró la utilidad de la estadística para dar información sobre el tipo de instancia de un determinado problema. La ecuación que define la *sparsity* de una matriz es

$$sp = \frac{n_0}{n^2} \text{ , donde}$$

n_0 : Número de entradas con valor de "0" en la matriz

Instancias clase (a)				Instancias clase (b)			
Instancia	$fd(B)$	$dd(A)$	sp	Instancia	$fd(B)$	$dd(A)$	sp
tai20a	64.90	67.02	0.015	nug30	112.48	52.75	0.316
tai25a	64.29	61.81	0.016	tho30	137.86	59.25	0.484
tai30a	63.21	58.00	0.013	tho40	155.54	53.20	0.585
tai35a	61.57	61.64	0.010	sko42	108.48	51.96	0.292
tai40a	60.23	63.10	0.009	sko49	109.38	51.55	0.304
tai60a	60.86	61.41	0.011	sko56	110.53	51.46	0.305
tai80a	60.38	59.22	0.009	sko64	108.38	51.18	0.308
rou20	64.43	65.65	0.010	sko72	107.13	51.14	0.299

Tabla 1. Resultados de las instancias seleccionadas en Stützle [1997] para las clases (a) y (b)

En las tablas 1 y 2 están dados los valores de $fd(B)$, $dd(A)$ y la *sparsity* de las instancias seleccionadas en Stützle [1997] para las clases (a) y (b) y las clases (c) y (d) respectivamente. En general, las instancias de las dos últimas clases tienen los mayores valores de dominancia y *sparsity*, reflejados aún más en la matriz de flujos. Por otro lado, las instancias de la clase (a) tienen valores más altos que los de las instancias de la clase (a).

Instancias clase (c)				**Instancias clase (d)**			
Instancia	$fd(B)$	$dd(A)$	*sp*	Instancia	$fd(B)$	$dd(A)$	*sp*
bur26a	274.95	15.09	0.223	tai20b	333.23	128.25	0.410
bur26b	274.95	15.91	0.223	tai25b	310.40	87.02	0.387
bur26c	228.40	15.09	0.257	tai30b	323.91	85.20	0.432
bur26d	228.40	15.91	0.257	tai35b	309.62	78.66	0.524
bur26e	254.00	15.09	0.312	tai40b	317.22	66.75	0.503
bur26g	279.89	15.10	0.211	tai50b	313.91	73.44	0.548
chr25a	57.97	424.27	0.883	tai60b	317.82	76.83	0.548
els19	531.02	52.10	0.637	tai80	323.17	64.05	0.552
kra30a	149.98	49.22	0.6	tai100	321.34	80.42	0.552
kra30b	149.98	49.99	0.6				
ste36a	400.30	55.65	0.707				
ste36b	400.30	100.79	0.707				

Tabla 2. Resultados de las instancias seleccionadas en Stützle [1997] para las clases (c) y (d)

Capítulo III

ALGORITMOS EMPLEADOS PARA ATACAR PROBLEMAS NP-HARD

Debido a su dificultad y a la enorme importancia práctica de los problemas de optimización combinatoria, un gran número de algoritmos han sido propuestos para resolverlos. Estos algoritmos se clasifican en: *exactos* o *completos* y *aproximados* o *heurísticos*. Los algoritmos exactos garantizan encontrar para toda instancia de tamaño finito una solución óptima en un tiempo finito de corrida [24]. Esto algoritmos incluyen entre otros, técnicas cómo: procesos de vuelta atrás (*backtracking*), ramificación y poda (*branch and bound*) y programación dinámica. Sin embargo, para problemas COP que son *NP*-hard no existe algoritmo que lo resuelva en tiempo polinomial y los métodos exactos podrían requerir de tiempos de corrida exponencial en el peor de los casos (Blum y Roli [2003]), que lleva a tiempos computacionales demasiado altos para propósitos prácticos. Esta es la razón que ha llevado al uso de los algoritmos aproximados en los últimos treinta años. En ellos, se sacrifica la exactitud (garantía de encontrar la solución óptima) por la eficiencia computacional (encontrar buenas soluciones -cercanas al óptimo- en un tiempo significativamente reducido).

Este trabajo se reduce a la aplicación de un algoritmo metaheurístico para resolver el problema QAP. Por tal razón se hará énfasis en los algoritmos heurísticos del que se basan las técnicas metaheurísticas conocidas anteriormente como heurísticas modernas.

3.1. ALGORITMOS HEURÍSITCOS

Existen otras razones para preferir los métodos heurísticos a los exactos además de la resolución de problemas *NP* -hard. Entre ellas se destacan las siguientes:

- El problema es de naturaleza tal que se desconoce algún método exacto para solucionarlo.
- Cuando se conoce algún método para resolver el problema, éste es computacionalmente costoso.
- Son más flexibles que los exactos, permiten por ejemplo, la incorporación de condiciones difíciles de modelar.
- Se pueden utilizar como parte de un procedimiento global que garantiza el óptimo de un problema. Existen dos posibilidades de uso:

    - Proporcionan una buena solución inicial de partida.
    - Participan en un paso intermedio del procedimiento, como por ejemplo la mejora de soluciones iniciales generadas aleatoriamente.

Los algoritmos heurísticos dependen en gran medida del problema concreto para el que han sido diseñados. Es decir, el modelamiento del algoritmo es específico del problema a solucionar y aunque en general pueden ser aplicadas a otros problemas deben particularizarse en cada caso.

Existen muchos métodos heurísticos de diversa naturaleza, lo que hace complicado dar una clasificación completa. Sin embargo, los heurísticos más conocidos pueden ubicarse dentro de cuatro categorías: los de *descomposición*, los *inductivos*, los de *reducción*, los *constructivos* y los de *búsqueda local*. Aunque todos estos métodos han contribuido a ampliar los conocimientos hacia la solución de los problemas, son

los constructivos y los de búsqueda local los que constituyen la base de los métodos metaheurísticos.

Algoritmo 1 Greedy construction heuristic

$S_p =$ solución vacía
While S_p no complete la solución **do**
  $e =$ componente greedy (S_p)
  $S_p = S_p \otimes e$
end
return S_p
end Greedy construction heuristic

Algoritmos constructivos: Construyen soluciones de forma incremental. Empiezan con una solución vacía e iteratívamente adhieren componentes hasta completarla. En el caso más sencillo los componentes son adheridos en orden aleatorio. Entre las técnicas más conocidas se encuentran las heurísticas de construcción voraz (*Greedy construcción heuristics*), ver Algoritmo 1 página 18. En cada iteración adhieren el componente con el máximo beneficio medido por alguna información heurística basada en reglas probabilísticas. La desventaja de este tipo de técnicas es que sólo generan un número muy limitado de soluciones. Además, las decisiones tomadas en las primeras iteraciones del proceso de construcción restringen directamente las posibilidades (posibles actividades a adherir en la solución parcial) en las iteraciones finales, determinando casi siempre movidas muy pobres en la etapa final de la construcción de la solución. Por este motivo, no existe garantía de que las soluciones que arroje sean óptimas con respecto a pequeños cambios a nivel local [35]. Para ello ha sido propuesto como mejora típica, refinar la solución obtenida con algún algoritmo de búsqueda local.

Algoritmos de búsqueda local: Empiezan con una solución inicial completa y tratan de encontrar una mejor solución en el vecindario de la solución actual. La versión

más sencilla son los algoritmos de mejora iterativos (*iterative improvement*), ver Algoritmo 2 página 19: si en el vecindario de la solución actual s se encuentra una solución mejor s', ésta reemplaza la solución actual y se continúa la búsqueda a partir de s'; si no se encuentra una mejor solución en el vecindario de la solución actual, el algoritmo termina en un óptimo local. La elección de la estructura apropiada del vecindario es uno de los aspectos cruciales de su desempeño y es específica del problema a ser resuelto. La estructura del vecindario define el conjunto de soluciones que pueden ser alcanzadas desde s en un sólo paso del algoritmo. Un ejemplo de ella para problemas como el TSP y el QAP es la $k-opt$ en la que las soluciones del vecindario difieren por al menos k arcos. Para definir completamente este tipo de algoritmos se necesita un esquema de examinación del vecindario que determina la forma de seleccionarlo y las soluciones dentro de él que reemplazarán a la solución actual. En el caso de los algoritmos de mejora iterativa la regla se conoce con el nombre de *pivoting rule*, dos ejemplos de ella son: la regla *best-improvement* que elige la solución del vecindario con la mayor mejora de la función objetivo y la regla *first-improvement* que toma la primera solución que mejore la función objetivo en la vecindad de la solución actual. Las desventajas de estos algoritmos es que quedan atrapados fácilmente en óptimos locales y dependen fuertemente de la solución inicial.

Algoritmo 2 Iterative Improvement ($S \in S$)

```
S' = Mejorar( S )
While S' ≠ S do
  S = S'
  S' = Mejorar( S )
  end
  return S
end Iterative improvement
```

Con el propósito de obtener mejores resultados a los alcanzados por los métodos heurísticos, en las últimas dos décadas ha surgido una nueva clase de algoritmos aproximados llamados comúnmente metaheurísticas. Las metaheurísticas o métodos metaheurísticos sirven como marco general para guiar la construcción de soluciones y/o la búsqueda local en las distintas heurísticas que la constituyen. Osman y Kelly [1995] introducen la siguiente definición: "Los procedimientos metaheurísticos son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria[5], en los que los heurísticos clásicos[6] no son efectivos. Los Metaheurísticos proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los mecanismos estadísticos".

De esta forma los métodos metaheurísticos se sitúan conceptualmente "por encima" de los heurísticos en el sentido que guían el diseño de éstos. Así mismo, a diferencia de los métodos heurísticos, un método metaheurístico puede aplicarse a diversos problemas de optimización combinatoria con pocos cambios significativos para cada problema.

3.2. METAHEURÍSTICAS

El término *metaheurístico* fue introducido por primera vez en Glover [1986] y deriva de la descomposición de dos palabras griegas, *heurístico* que proviene del verbo *heuriskein* que significa "encontrar", y el prefijo *meta* que significa "más allá, en un nivel superior". Antes de que el término fuera ampliamente adoptado, las metaheurísticas eran reconocidas bajo el nombre de *heurísticas modernas*. Varias propuestas se han hecho en los últimos años para definirla, pero ninguna ha sido

[5] De hecho las metaheurísticas son ampliamente reconocidas como una de las mejores aproximaciones para atacar problemas de optimización combinatoria [35].

[6] Son los mismos métodos heurísticos sólo que Oman y Kelly [1995] hacen la diferencia entre los clásicos para referirse a los heurísticos, y los complejos y modernos para referirse a los metaheurísticos.

aceptada en común acuerdo. Aquí se dan dos de las definiciones que mejor capturan la descripción y el propósito de una metaheurística:

"Una metaheurística es un proceso maestro que guía y modifica las operaciones de heurísticas subordinadas para producir eficientemente soluciones de alta calidad. Ellas pueden manipular una única solución completa (o incompleta) o una colección de soluciones en cada iteración. Las heurísticas subordinadas pueden ser procedimientos de alto o bajo nivel, o simplemente métodos de búsqueda local o de construcción." [VoB et al. 1999].

"Una metaheurística es un conjunto de conceptos que pueden ser usados para definir los métodos heurísticos que pueden ser aplicados a un amplio grupo de diferentes problemas. En otras palabras, una metaheurística puede ser vista como un sistema algorítmico general que puede ser aplicado a diferentes problemas de optimización con relativamente pocas modificaciones para que sea adaptado a un problema específico.[7]" [Metaheuristics Network Website 2000].

La consideración de todas las definiciones del término ha permitido resumir las características comunes a toda metaheurística en los atributos fundamentales que se mencionan a continuación:

- Son estrategias que guían procesos de búsqueda de soluciones a un nivel determinado.
- Su éxito radica en explorar efectivamente el espacio de búsqueda con el fin de obtener soluciones óptimas o pseudo óptimas (cercanas al óptimo) a un bajo costo computacional.

[7] Cuando se habla de la metaheurística xxx, donde xxx representa el nombre de ella, se está refiriendo al esquema-procedimiento general. Mientras que el término algoritmo xxx, es usado para indicar cualquier instancia genérica de la respectiva metaheurística. Las instancias son algoritmos aplicados a problemas específicos que pueden tener varias versiones de acuerdo al número de problemas que hayan sido resueltos con ellos.

- Las técnicas que la constituyen van desde una simple heurística de búsqueda local hasta complejos procesos de aprendizaje.
- Son algoritmos aproximados y generalmente no determinísticos (de ahí el uso de mecanismos probabilísticos).
- No son específicas para cada problema.
- Hacen uso del dominio específico de conocimientos en la forma de heurísticas que son controladas por una estrategia (algoritmo) de nivel superior.
- Actualmente las más avanzadas hacen uso de la experiencia de búsqueda (representado por algún tipo de memoria) para guiar la búsqueda.

El nivel de búsqueda en una metaheurística se refiere a la concentración de la búsqueda en una región particular del espacio y/o a permitir la búsqueda de áreas distintas. De esta forma, la búsqueda desarrollada por la metaheurística debe ser lo suficientemente "ingeniosa" como para conducir la técnica a la exploración intensiva de áreas en el espacio de búsqueda con soluciones de alta calidad, y a moverse a áreas inexploradas cuando sea necesario [4]. Estos dos conceptos comprenden lo que hoy en día se conocen como las estrategias de *intensificación* y *diversificación*. Y son consideradas las fuerzas que dirigen las metaheuríticas hacia un alto desempeño.

Uno de los temas de investigación más promisorios en este campo es la hibridación de las metaheurísticas, de hecho algunos de los algoritmos metaheurísticos más exitosos de los últimos años son hibridizaciones, como lo es el algoritmo HAS-QAP aplicado en este trabajo.

3.2.1. Clasificación de las metaheurísticas

Existen diferentes formas de clasificar y describir las técnicas metaheurísticas. Las posibles clasificaciones dependen de la característica seleccionada para diferenciarlas.

Aquí sólo se describirán las características de acuerdo al criterio general que han empleado varios autores para clasificar a los algoritmos metaheurísticos.

Bio-inspirados vs. no bio-inspirados: Esta clasificación toma en cuenta la fuente de inspiración del algoritmo. Entre los métodos bio-inspirados encontramos entre otros, los *Algoritmos genéticos* (GA), los *Algoritmos hormigas* (Ant Algorithms) y los de *Optimización por colonia de hormigas* (ACO). Por el lado de los no bio-inspirados están los algoritmos de *Búsqueda Tabú* (TS) y los de *Búsqueda local iterativa* (ILS), etc. Blum y Roli [2003] consideran esta clasificación como la menos significativa por dos razones. La primera es que los recientes algoritmos híbridos no encajan en ninguna de las clases o puede que encajan en ambas al mismo tiempo. En segundo lugar, en ocasiones es difícil atribuir claramente un algoritmo a una de dos las clases.

Basados en la población vs. basados en un sólo punto de búsqueda: Hace relación al número de soluciones usadas al mismo tiempo, es decir, si el algoritmo trabaja sobre una población de soluciones o sobre una única solución en cada iteración. Los algoritmos basados en la población proveen una forma conveniente para la exploración del espacio de búsqueda. Aún así, el desempeño del algoritmo depende en últimas de la manera en que la población es manipulada (Birattari, Paquete, Stützle y Varrentrapp [2001]).

Métodos de trayectoria vs. métodos discontinuos: Permite la distinción entre las metaheurísticas que siguen una única trayectoria de búsqueda correspondiente a un camino cerrado en el grafo de la vecindad y aquellas en las que es posible explorar nuevos caminos (dar saltos) en el grafo de la vecindad. Algoritmos que trabajan sobre únicas soluciones pueden ser considerados también métodos de trayectoria, que abarcan a algoritmos tales como: TS, ILS y el algoritmo de *Búsqueda variable de la vecindad* (VLS). Por el contrario algoritmos basados en una población desarrollan procesos de búsqueda que describen la evolución de un conjunto de puntos

(soluciones iniciales) en el espacio de búsqueda. Cada una de estas soluciones corresponde a saltos en el grafo de la vecindad. Estos algoritmos, siguen en general un camino discontinuo con respecto al grafo de la vecindad usado por los algoritmos de búsqueda local basados en una única solución. Dos de ellos son GA y ACO.

Función objetivo dinámica vs. función objetivo estática: Clasifica los algoritmos metaheurísticos de acuerdo a la manera en que ellos hacen uso de la función objetivo. Los algoritmos con función objetivo estática mantienen inalterable la función objetivo dada en la representación del problema, durante todo el proceso de búsqueda. Mientras que los algoritmos con función objetivo dinámica, tal como el algoritmo de *Búsqueda local guiada* (GLS), la modifican durante la búsqueda con la idea de escapar de óptimos locales. La función objetivo es alterada mediante la incorporación de información recolectada durante el proceso de búsqueda.

Única estructura de búsqueda vs. varias estructuras de búsqueda: La mayoría de las metaheurísticas utilizan algoritmos de búsqueda local aplicados sobre una única estructura del vecindario que define el tipo de movidas. Este es el caso de los algoritmos de *Enfriamiento simulado* (SA) y TS. Otras metaheurísticas en cambio, emplean algoritmos de búsqueda local aplicados a varias estructuras del vecindario. El algoritmo ILS por ejemplo, usa dos estructuras diferentes N y N'. La búsqueda local empieza con la estructura N hasta que obtiene un óptimo local y luego usa la segunda N' hasta alcanzar un óptimo local con esta última. El algoritmo de *Búsqueda variable de la vecindad* (VNS) introduce la idea del intercambio sistemático de k estructuras diferentes.

Métodos con memoria vs. métodos sin memoria: Se refiere al uso que hacen las metaheuríticas de la experiencia de la búsqueda, para influenciar la dirección futura que tomará la misma, esto es, si usan memoria o no. Los algoritmos sin memoria

desarrollan un proceso de markov como la única información que determina la próxima acción en el estado actual del proceso de búsqueda, entre ellos los algoritmos SA y GRASP. Las memorias en las metaheurísticas son consideradas usualmente de dos tipos: *memorias de corto plazo* y *memorias de largo plazo*. Las primeras son usadas para impedir visitar de nuevo elementos de la solución parcial y eliminar los ciclos. Las segundas, se usan para intensificar y diversificar la búsqueda (ahí es donde aplican las estrategias de intensificación y diversificación).

Algoritmo 3 Procedimiento AMP

1. **Inicializar** la memoria
2. **Repetir** hasta que un criterio de terminación sea satisfecho:
   2a. **Construir** una solución inicial usando la información contenida en la memoria
   2b. **Mejorar** la solución inicial con un algoritmo de búsqueda local
   2c. **Actualizar** la memoria

El concepto *Programación adaptativa de la memoria* (AMP) [25][8] ha sido propuesto recientemente para agrupar un número de metaheurísticas que tienen en común el uso de memoria y algoritmos de búsqueda local. En las metaheurísticas, el término AMP se asemeja a un proceso de aprendizaje. El esquema general de un algoritmo AMP es el siguiente, ver Algoritmo 3 página 25: una nueva solución o conjunto de soluciones del problema es generada creando en primer lugar una solución o conjunto de soluciones iniciales empleando la información contenida en la memoria que luego refina (las mejora) con un algoritmo de búsqueda local. Por último, la solución o conjunto soluciones refinadas son usadas para actualizar la memoria y el proceso es repetido hasta que se cumpla un criterio de terminación. A partir de este esquema se pueden distinguir diversas técnicas metaheurísticas de acuerdo al tipo de implementación de la memoria, al proceso de construcción de soluciones iniciales, al método de mejora de las soluciones y a la forma de actualizar la memoria.

[8] En [25] se proponen, comparan y analizan metaheurísticas con uso de memoria, aplicadas al problema QAP, una de ellas es la metaheurística HAS-QAP.

Característica	SA	TS	GA	ACO[9]	ILS	GLS
Bio-Inspirados	√	¬	√	√	¬	¬
Población	¬	¬	√	√	¬	¬
Trayectoria	√	√	¬	¬	¬	√
Función objetivo dinámica	¬	∃	¬	¬	¬	√
Única estructura	√	√	∃	√	¬	√
Memoria	¬	√	∃	√	∃	√

Tabla 3. Caracterización de algunas metaheurísticas con base al criterio general de clasificación de la sección 3.2.1

En la tabla 3 se caracterizan varios de los métodos metaheurísticos mencionados de acuerdo del criterio general de clasificación. Los símbolos √, ¬ y ∃ indican presencia, no presencia y presencia parcial de la característica.

3.2.2. Intensificación y diversificación

A pesar de la relevancia de estas dos estrategias en el desempeño de las metaheurísticas, sus descripciones son genéricas y específicas de cada metaheurística. La manera en que son implementadas depende del objetivo particular que hay detrás de cada técnica (Blum y Roli [2003]). Además, de la combinación de estrategias que hacen parte de diferentes metaheurísticas se pueden obtener algoritmos metaheurísticos híbridos.

Durante la etapa de intensificación la búsqueda se enfoca en la examinación del vecindario de soluciones élite (mejores soluciones encontradas hasta el momento). En la etapa de diversificación el proceso de búsqueda es dirigido hacia regiones no exploradas con el fin de obtener soluciones que difieran significativamente en

[9] El algoritmo HAS-QAP recoge algunas de las características de la metaheurística ACO, tales como su inspiración biológica, el uso de memoria y la manipulación de una población de soluciones basada en una única estructura del espacio de búsqueda.

diferentes formas, de las encontradas hasta el momento. En algunas instancias, la intensificación puede ser concebida como una estrategia de plazo intermedio y la diversificación como una estrategia que aplica a consideraciones (evadir la estancación del algoritmo en óptimos locales) que salen a relucir en el largo plazo (Glover y Laguna [1997])[10].

De la medida en que el diseño de una metaheurística provea el balance adecuado entre la intensificación y la diversificación, depende su éxito resolviendo un determinado problema de optimización.

Un punto de vista unificado sobre las estrategias de intensificación y diversificación fue dado en [4] con el nombre de *marco I&D*[11]. Este marco pretende facilitar el entendimiento de las similaridades y diferencias entre metaheurísticas en función de éstas estrategias. Para ello definen a un *componente I&D* como cualquier algoritmo (heurística) o componente funcional contenido en la metaheurística que tenga un efecto en la intensificación y/o diversificación del proceso de búsqueda. Lo anterior implica la existencia de componentes con ambas funciones (intensificación y diversificación). Los componentes I&D se pueden clasificar en *básicos* o *intrínsecos* y *estratégicos*.

Componentes básicos o intrínsecos: Son definidos por las ideas básicas de una metaheurística. Hacen relación a los componentes que constituyen la metaheurística base que reúne todos los posibles algoritmos (también metaheurísticas, pero más específicas) con características propias de ella (ideas básicas de la metaheurística específica). En el caso de la metaheurística ACO que agrupa entre otros, a los algoritmos *Sistema de hormigas* (AS) y *Sistema de colonia de hormigas* (ASC), comparten como componente básico de toda ACO, la *actualización de la matriz de*

[10] El algoritmo HAS-QAP utiliza ésta concepción en la implementación del *mecanismo de diversificación*.
[11] La letra *I* hace referencia a la estrategia de intensificación y la letra *D* a la estrategia de diversificación.

feromonas que funciona básicamente como mecanismo de intensificación. La actualización de feromonas guiada por el valor de la función objetivo actualiza con mayor cantidad de feromonas, a aquellos componentes (actividades) que hacen parte de la mejor solución encontrada por la búsqueda en cada iteración.

Componentes estratégicos: Los constituyen todas aquellas técnicas y estrategias de mediano y largo plazo que el diseñador adhiere a la metaheurística base para mejorar su desempeño. Muchas veces han sido desarrolladas originariamente en metaheurísticas específicas y más luego generalizadas.

En las metaheurísticas se identifican también componentes y estrategias que cumplen doble función, envuelven un componente de intensificación y uno de diversificación al tiempo. En el componente actualización de la matriz de feromonas de la metaheurística ACO el componente de diversificación esta representado por la función de *evaporación del feromona*.

3.2.3. Hibridización de las metaheurísticas

Para Blum y Roli [2003] existen tres formas de hibridización:

Intercambio de componentes (heurísticas) entre metaheurísticas[12]**:** Consiste en la inclusión de componentes de una metaheurística dentro de otra. El más común se refiere al uso de métodos de trayectoria en métodos basados en la población. Dos de los ejemplos más exitosos son los híbridos formados por las metaheurísticas GA y ACO (métodos basados en la población) con algoritmos de búsqueda local (encontrados en los métodos de trayectoria).

[12] Esta forma de hibridización es la utilizada en el algoritmo HAS-QAP y en todos los algoritmos ACO aplicados al problema QAP.

Búsqueda cooperativa: Consiste de una búsqueda desarrollada por varios algoritmos que intercambian información acerca de los estados, subproblemas, soluciones o características de otros espacios de búsqueda. Por lo general, son algoritmos de búsqueda ejecutados en paralelo con variados niveles de comunicación. Estos algoritmos pueden ser diferentes (diferentes metaheurísticas), o pueden ser: instancias del mismo algoritmo trabajando en diferentes modelos del problema o el mismo algoritmo corrido para diferentes parámetros.

Integración de metaheurísticas y métodos sistemáticos: Algoritmos híbridos generados recientemente con esta técnica han resultado ser muy efectivos cuando son aplicados a instancias de la vida real. Hay tres maneras de realizar la integración entre metaheurísticas (generalmente métodos de trayectoria) y métodos sistemáticos (como *Programación restringida* (CP) y *Árbol de búsqueda*).

- La metaheurística y el método sistemático se aplican consecutivamente. La metaheurística se corre para el problema determinado con el fin de generar soluciones iniciales que más luego serán usadas como información heurística por el método sistemático.
- La metaheurística hace uso del método sistemático para explorar eficientemente la vecindad de las soluciones.
- Introduce conceptos y estrategias de cada clase de algoritmo dentro de otro. Un ejemplo de ello es el uso de de los conceptos de lista tabú y criterio de aspiración (dados en la metaheurística TS) para definir la lista de nodos abiertos (nodos cuyos hijos no han sido explorados aún) en el método sistemático.

3.3. ARQUITECTURA MULTI-NIVEL DE LAS METAHEURÍSTICAS

Los conceptos que llevan a la definición de un tipo de arquitectura para las metaheurísticas son de hecho los que lleva implícito como algoritmo aproximado:

- Una metaheurística es un marco general de referencia para el diseño de nuevas instancias.
- Una metaheurística es un proceso maestro que guía y modifica las operaciones de heurísticas subordinadas para producir eficientemente soluciones de alta calidad.

Las metaheurísticas pueden ser descritas entonces como la interacción de agentes (componentes-heurísticas del algoritmo) cada uno con una función específica, mediante un proceso implícito que guía su comunicación y sincronización en la búsqueda de soluciones de alta calidad enmarcado dentro de un sistema (marco general) que simula su medio ambiente.

De esta forma, la arquitectura de una metaheurística brinda un modelo para la descripción de su marco. El tipo de arquitectura que se utilizará de ahora en adelante para la descripción de las metaheurísticas que se mencionen a lo largo del texto será la *Arquitectura multi-nivel* (MLA) (ver modelo en figura 4), o *multi-agente* basado en la agrupación de agentes con la misma función en común, propuesta por Milano y Roli [2001]. MLA esta compuesta por cuatro niveles, cada uno de ellos reúne un conjunto de agentes especializados en una tarea específica (función). Sin embargo, sólo se trabajará con los tres primeros niveles que constituyen la base de toda metaheurística. Las funciones de los agentes en el último nivel (nivel 3) hacen referencia a las funciones más importantes encontradas en cada uno de los niveles anteriores, por esa razón, son considerados agentes de razonamiento y aprendizaje.

El tipo de agentes encontrados en cada uno de los tres primeros niveles de MLA se define en la siguiente clasificación.

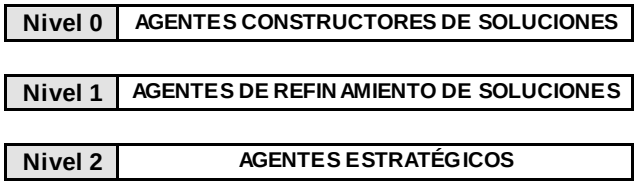


Figura 4. Arquitectura multi-nivel de tres niveles

3.3.1. Niveles de MLA

Nivel 0: A él pertenecen los *agentes constructores de soluciones*, cuya función es producir soluciones iniciales que más luego serán utilizadas por niveles superiores. Los algoritmos que pueden ser implementados por los agentes pueden ser de inicialización aleatoria, construcción voraz, construcción probabilística (como aquellos basados en feromonas), etc. Se les considera agentes reactivos porque su comportamiento es guiado por información heurística que es cambiada dinámicamente por los agentes de niveles superiores.

Nivel 1: Lo conforman los *agentes de refinamiento de soluciones*. Son agentes que implementan varios algoritmos de búsqueda que tratan de mejorar las soluciones iniciales recibidas del nivel 0. Estos agentes pueden estar constituidos por agentes independientes cada uno aplicando un algoritmo diferente o agentes cooperativos que intercambian información. El concepto de memoria de corto plazo es asociado a éste nivel en el que los agentes hacen uso su pasado más reciente para intensificar la búsqueda o escapar de óptimos locales. Las soluciones generadas en este nivel brindan información útil acerca de de la estructura de las soluciones óptimas o subóptimas que será utilizada después por el nivel 2 para guiar la búsqueda hacia ellas.

Nivel 2:.Entre las tareas que pueden ser realizadas por los agentes del nivel 2 están la memorización de las mejores soluciones encontradas, sugerir cuales bloques de construcción tienen más probabilidad de estar incluidos en la solución óptima, sugerir qué regiones del espacio de búsqueda son más promisorias y seguir examinando regiones de soluciones ya exploradas, dadas por las soluciones que genera el nivel 1. Estas tareas están dirigidas a la intensificación y diversificación de la búsqueda o a proveer nuevos bloques de construcción de soluciones. Al nivel 2 pertenecen los agentes estratégicos (componentes I&D): *agentes analizadores de bloques de construcción* encargados de extraer soluciones parciales que hacen parte de las buenas soluciones, y los *agentes analizadores de espacios de búsqueda* que examinan el espacio de búsqueda para hallar regiones promisorias no exploradas. El concepto de memoria de largo plazo se implementa en este nivel: la historia completa de la búsqueda se utiliza para intensificarla y diversificarla. Algunos de los algoritmos y técnicas implementadas por estos agentes son: algoritmos de comparación directa entre soluciones y mecanismos estadísticos.

De manera generalizada un algoritmo metaheurístico modelado con MLA consiste en la implementación secuencial y sincronizada de los niveles: el nivel 0 genera una solución o conjunto de soluciones iniciales, el nivel 1 las mejora y el nivel 2 determina la próxima región a ser explorada.

3.3.2. Interconexión entre los niveles de MLA

En el modelo MLA todas las comunicaciones directas entre niveles deben ser representadas. La presencia de niveles hace explícito la presencia de interconexiones que muestren el flujo de información.

Es importante recalcar que diferentes agentes pueden trabajar a la vez en el mismo nivel. Estos agentes usan diferentes representaciones de la función característica del nivel pero se comunican por medio de agentes que filtran y trasladan la información

(agentes del nivel 3). La comunicación entre este tipo de agentes ha sido denominada por Milano y Roli [2001] como *cooperación horizontal*. Y a la comunicación entre agentes de diferentes niveles *cooperación vertical*.

Capítulo IV

ALGORITMOS HORMIGAS

Las colonias de hormigas y en general las colonias de insectos sociales, son sistemas distribuidos que a pesar de la simplicidad de sus individuos presentan una organización social altamente estructurada. Como resultado de esta organización, las colonias de hormigas pueden llevar a cabo tareas complejas que en algunos casos sobrepasan las capacidades de una única hormiga. El estudio del comportamiento de las colonias de este tipo de especies y de las capacidades de su propia organización resulta interesante para científicos computacionales, porque proveen modelos de organización distribuida que sirven de fuente de inspiración para desarrollar algoritmos en el campo de la inteligencia artificial (AI) y la inteligencia de enjambres (SI) para resolver difíciles problemas de optimización.

Los algoritmos hormigas comprenden todos aquellos modelos derivados de la observación de las colonias de hormigas. La idea principal de estos algoritmos es que el principio de *auto-organización* [6, 10, 11, 14, 16] que lleva al comportamiento coordinado de las hormigas reales para el logro de una tarea, sea explotado para coordinar poblaciones de agentes artificiales que colaboran entre si para resolver problemas computacionales. Ejemplos de estas tareas son la división de labores y el transporte cooperativo de alimentos. En todos ellos, las hormigas coordinan sus actividades a través de *stigmergy* [6, 14], una forma de comunicación indirecta guiada por modificaciones en el medio ambiente. Los biólogos han demostrado que la comunicación indirecta *stigmergy,* casi siempre es suficiente para que las hormigas y en general, los insectos sociales alcancen su propia organización (Bonabeau, Dorigo y Theraulaz [2000]). De esta forma, detrás de todo algoritmo hormiga esta el uso de

una forma de comunicación indirecta (*stigmergy artificial*) para coordinar las poblaciones de agentes artificiales.

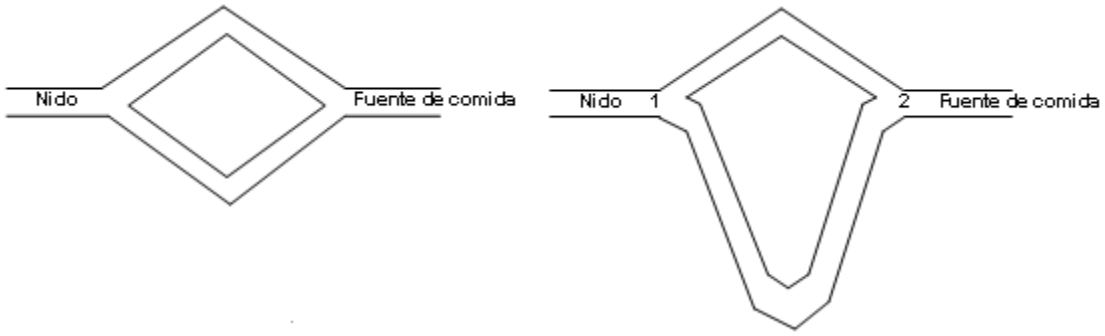
Uno de los comportamientos más importantes e interesantes que exhiben las colonias de hormigas es el de búsqueda, utilizado en la consecución y transporte de alimento y que explica cómo las hormigas pueden encontrar el camino más corto entre la fuente de comida y su nido.

4.1. DESCRIPCIÓN DEL PROCESO DE BÚSQUEDA DEL CAMINO MÁS CORTO

Mientras caminan de la fuente de comida al nido y viceversa, las hormigas depositan en el suelo una sustancia química llamada *feromona*, formando un rastro de feromona. Las hormigas pueden oler el feromona y de esta manera, cuando eligen su camino tienden a tomar, en probabilidad, los caminos marcados por fuertes concentraciones de feromona. El rastro de feromona lleva a las hormigas a encontrar su camino de regreso a la fuente de comida o al nido. Así mismo, puede ser usado por otras hormigas para dar con la ubicación de la fuente de comida encontrada por las anteriores. Se ha demostrado experimentalmente que cuando hay más caminos disponibles del nido a una fuente de comida, una colonia de hormigas puede ser capaz de explotar el rastro de feromona dejado por cada una de ellas para descubrir el camino más corto [15]. El rastro de feromona y el comportamiento de las hormigas durante este proceso han sido investigados en experimentos controlados, algunos de ellos diseñados por Deneubourg y sus colegas entre 1989 y 1990. Deneubourg usó un puente doble (con dos trayectos) para conectar el nido a la fuente de comida y corrió los experimentos variando la longitud de los trayectos.

Trayectos con igual longitud: El primer experimento suponía ambos trayectos de igual longitud. Dos fases sucesivas toman lugar en este proceso de búsqueda [6]. Al

principio aparece una *fase no-coordinada* (caracterizada por la elección aleatoria de caminos): las hormigas son totalmente libres de elegir entre ambos trayectos, le sigue una *fase de cooperación* (guiada por la concentración de rastros de feromonas-comunicación stigmergic-): rápidamente la mayoría de las hormigas terminan utilizando un determinado trayecto aún cuando no existe razón para preferir un trayecto por encima del otro. Las razones de este resultado fueron los siguientes: cuando el proceso empieza, no existe feromona en ninguno de los trayectos, de ahí que las hormigas no tienen preferencia y seleccionan con la misma probabilidad cualquiera de ellos. Sin embargo, debido a fluctuaciones aleatorias, unas cuantas hormigas eligen un trayecto por encima del otro, que como consecuencia, tendrá mayor concentración de feromona. Esta alta concentración conlleva a más hormigas eligiendo el mismo trayecto hasta que la mayoría de las ellas converge hacia él. Este comportamiento colectivo que lleva a la convergencia hacia un sólo trayecto es una forma del comportamiento auto-organizado que emerge en las hormigas, conocido con el nombre de mecanismo *autocatalítico* [6, 8, 11, 14]: entre más hormigas sigan un mismo trayecto, más atractivo se vuelve para que sea seguido (más probabilidades de ser elegido). El mecanismo es caracterizado por una *retroalimentación positiva* (positive feedback) [11, 14], donde la probabilidad con la que una hormiga elige un trayecto incrementa con el número de hormigas que lo hayan elegido previamente. Es además, un ejemplo de comunicación stigmergic: las hormigas coordinan sus actividades, explotando comunicaciones indirectas entre ellas guiadas por las modificaciones del medio ambiente en el que ellas se mueven (concentraciones en menor o mayor grado sobre los trayectos dan aviso indirecto a otras hormigas sobre la preferencia de un trayecto por encima de otro).



a) **b)**

Figura 5. Representación gráfica del puente doble utilizado en los experimentos. a) Trayectos con longitudes iguales y b) Trayectos con longitudes diferentes

Trayectos con diferente longitud: Este segundo experimento asumió que la longitud de un trayecto fuera dos veces la longitud del otro. En la mayoría de las corridas que se hicieron de este caso, después de un tiempo, las hormigas decidían usar sólo el trayecto más corto. Al igual que en el primer experimento, al inicio las hormigas se decidían por alguno de los trayectos en forma aleatoria. Luego, era esperado que en promedio, la mitad de las hormigas eligieran el trayecto más corto y la otra mitad el trayecto más largo, aunque las fluctuaciones aleatorias podían favorecer a un trayecto más que al otro. La principal diferencia con respecto al experimento anterior es que al haber un trayecto más corto que el otro, las hormigas que elegían el más corto eran las primeras en llegar a la fuente de comida y empezar su retorno al nido (Di Caro y Dorigo [1999]). Por eso, cuando ellas debían tomar de nuevo la decisión entre el trayecto más corto y el más largo, el alto nivel de feromona en el corto sesgaba su decisión en favor del trayecto más corto. De esta forma, el rastro de feromona empezó a acumularse más rápidamente sobre el trayecto más corto, que más tarde sería por todas las hormigas de la colonia debido al mecanismo autocatalítico. La influencia de las fluctuaciones aleatorias (fase de no-coordinación) en éste experimento, es mucho más reducida, y la obtención del camino más corto depende aún más de la fase de cooperación mediada por los mecanismos autocatalítico y la *evaluación implícita* [14, 15] de trayectos. La evaluación implícita de trayectos

(soluciones en las hormigas artificiales de los algoritmos hormigas) se refiere al hecho de que los trayectos cortos serán completados primero que los largos, y por tanto la acumulación de feromonas en ellos es más rápida. La combinación del mecanismo autocatalítico y la evaluación implícita de trayectos es efectiva, en la medida que un trayecto sea más corto que otro, entre más corto sea, más pronto es depositado el feromona por las hormigas en él, y más hormigas decidirán usarlo (Di Caro y Dorigo [1999]). Sin embargo, el inconveniente de la autocatalisis es que se debe tener cierto cuidado para evitar la convergencia prematura (*estancación*) [4, 6, 14]. Esta es la situación en la que alguna no tan buena trayectoria (no tan corta) absorbe la decisión de la población de hormigas debido a una situación no prevista (ya sea por que es un óptimo local, o porque las fluctuaciones iniciales aleatorias hacen que la trayectoria sea considerada la mejor, por todos los individuos de la población) que impide seguir explorando el espacio de búsqueda (otros posibles caminos entre el nido y la fuente de camino). La evaporación del feromona y los estados de transición estocásticas son los mecanismos de *retroalimentación negativa* [14] que evitan la estancación del proceso de búsqueda.

De esta forma, el proceso de búsqueda descrito a través de los experimentos de doble trayecto, es en sí un mecanismo de optimización distribuido en el que cada hormiga da sólo una pequeña contribución (Di Caro, Dorigo y Gambardella [1999]). Al principio, una única hormiga es capaz de encontrar una solución (encontrar un trayecto entre el nido y la fuente de comida), pero es únicamente el comportamiento colectivo de la colonia de hormigas, el que encuentra el trayecto más corto, mediante el uso de una forma de comunicación indirecta conocida como stigmergy que es guiada por los rastros de feromonas.

4.2. DEFINICIÓN Y CARACTERÍSTICAS DE LOS ALGORITMOS HORMIGA

Un algoritmo hormiga es definido informalmente como un sistema multiagente[13] inspirado por la observación del comportamiento de colonias de hormigas reales explotando stigmergy. En los algoritmos hormigas los agentes son llamados *hormigas artificiales* o simplemente *hormigas*, y la coordinación entre las hormigas se logra a través de un mecanismo de comunicación indirecta conocido como stigmergy (Bonabeau, Dorigo, y Theraulaz [2000]). Dentro del campo de la inteligencia artificial (IA), los algoritmos hormigas representan uno de las más exitosas aplicaciones de inteligencia de enjambres (SA) (Di Caro, Dorigo y Gambardella [1999]).

A partir del comportamiento de la colonia de hormigas observada en el experimento de doble trayecto fueron varios los modelos los que se formularon para definir un algoritmo hormiga. Los modelos iniciales describían a un algoritmo hormiga como una reproducción exacta del comportamiento de las hormigas reales desempeñado por hormigas artificiales. Por eso consideraron un tiempo continuo para el desarrollo de las actividades de las hormigas artificiales en la construcción de las soluciones. Más tarde se propuso un modelo discreto en el tiempo basado en el grafo con que podía ser representado el experimento de doble trayecto, pero al igual que los anteriores, dotaba a las hormigas artificiales únicamente con las capacidades que las hormigas reales explotaban en la búsqueda del camino más corto. Finalmente, el grafo propuesto para el experimento de doble trayecto fue generalizado para que fuera aplicable a los grafos complejos que representan a los problemas de optimización de mínimo costo como los problemas TSP y QAP.

[13] En la ciencia computacional un *Sistema multi-agente* (MAS) es un sistema compuesto de varios agentes que colectivamente son capaces de alcanzar metas que son difíciles de lograr por un agente individualmente, correspondiente a un sistema monolítico [wikipedia].

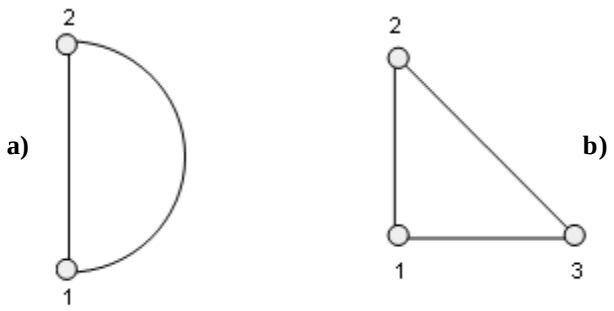


Figura 6. Grafos equivalentes a la representación gráfica del experimento de doble trayecto de la figura 5b. a) En este grafo, la longitud del trayecto más largo (curva) es r veces la longitud del arco más corto (recta). b) En este grafo, los arcos más cortos tienen longitudes iguales y la longitud del tercer arco es r veces la longitud de cualquiera de los anteriores. Aquí, el trayecto más largo es representado por una secuencia de arcos

Modelo estocástico y continuo en el tiempo: Gross et al. [1989] y Deneubourg et al. [1990] fueron los primeros en proponer un modelo que describía la dinámica del comportamiento de la colonia de hormigas observada en los experimentos de doble trayectoria. En un principio era un modelo estocástico sencillo y continuo en el tiempo que asumía que la cantidad de feromona en un trayecto era proporcional al número de hormigas que lo habían visitado en el pasado y por esa razón, la evaporación del feromona no fue considerado. En el modelo, las hormigas depositaban el feromona en el camino de ida a la fuente de comida como en el de su regreso al nido; este comportamiento era considerado necesario para la convergencia de la colonia de hormigas hacia el trayecto más corto. La observación real de las colonias de hormigas en los experimentos de doble trayecto ha confirmado que las hormigas que depositan feromona sólo cuando retornan al nido son incapaces de encontrar el camino más corto entre el nido y la fuente de comida.

Modelo estocástico y discreto en el tiempo basado en la construcción de soluciones en el grafo de representación del experimento de doble trayecto: En él

una población de hormigas artificiales mediante el uso de reglas probabilísticas basadas en el rastro de feromona artificial, se mueve en un grafo modelando el puente doble hasta encontrar el camino más corto entre los dos nodos correspondientes al nido y a la fuente de comida (ver figura 6). Los dos nodos están conectados por un arco corto y por uno largo que simulan los dos trayectos. Se asume que el tiempo es discreto, $t = (1,2,...)$ y que a cada unidad de tiempo la hormiga se mueve hacia la fuente de comida adhiriendo una unidad de feromona en los arcos que usa para llegar hasta allí. Las hormigas se mueven en el grafo eligiendo el arco a seguir de manera probabilística: $p_{is}(t)$ es la probabilidad de que una hormiga ubicada en el nodo i en el tiempo t elija el arco corto, y $p_{il}(t)$ la probabilidad de elegir el arco largo. Estas probabilidades son una función del rastro de feromona φ_{ia} que las hormigas en el nodo $i(i \in \{1,2\})$ encuentren en el arco $a \in \{s,l\}$, donde s y l denotan el arco corto y el largo respectivamente.

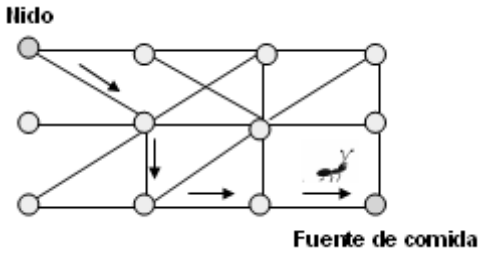


Figura 7. Representación de la hormiga artificial construyendo una solución en el grafo generalizado *G*

Modelo basado en la construcción de soluciones en un grafo generalizado para la aplicación de algoritmos hormigas a problemas de optimización de mínimo costo: El éxito en el diseño de los algoritmos hormigas radica en su aplicación a grafos mucho más complicados (ver figura 7) (que los del experimento de doble trayecto) que representan problemas difíciles de optimización con mínimo costo (o

máximo costo). En la figura se considera un grafo estático conectado $G = (N, A)$, donde N es el conjunto de $n = |N|$ nodos y A es el conjunto de arcos no dirigidos que los conectan; cada uno de ellos tiene un costo asociado. Dos nodos $i, j \in N$ son vecinos si existe un arco $(i, j) \in A$. Los puntos entre los cuales se establece el mínimo costo, son llamados nodo fuente (nido) y nodo destino (fuente de comida). Sin embargo, resolver este tipo de problemas en el grafo G no podía ser resuelto con hormigas artificiales siguiendo simplemente el comportamiento de las hormigas reales descrito anteriormente. El hecho de depositar feromonas tanto en los trayectos de ida a la fuente de comida como en los de su regreso al nido, hace que las hormigas mientras construyen la solución puedan generar ciclos [15]. Los ciclos tienden a hacerse más atractivos y las hormigas pueden quedar atrapadas en ellos. Aún cuando las hormigas pueden escapar de los ciclos, la distribución de los rastros de feromonas hace que los trayectos cortos no se favorezcan, y que el mecanismo que llevaba a las hormigas a elegir con mayor probabilidad los trayectos más cortos en el experimento de doble trayecto no funcione en la construcción de la solución en el grafo G. Por esta razón fue necesario extender las capacidades de las hormigas artificiales trabajando en el grafo, más allá de las adoptadas del comportamiento de las hormigas reales. Principalmente, fueron dotadas con un tipo de memoria limitada que almacena los trayectos que habían seguido hasta el momento y también los costos asociados a cada uno de ellos. A través de esta memoria las hormigas artificiales podían desarrollar una cantidad de comportamientos útiles para llevarlas a la construcción eficiente de soluciones de mínimo costo para los problemas de optimización. En la sección 4.3 se explica detalladamente las capacidades extras con las que fueron reforzadas las hormigas artificiales para que fueran capaces de resolver este tipo de problemas.

Los modelos considerados coinciden en ciertos aspectos básicos que son los que actualmente caracterizan a los algoritmos hormigas [15]. Ellos son:

- Una colonia de hormigas artificiales cooperando entre ellas
- Comunicación stigmergic entre hormigas basada en el rastro de feromonas
- Una secuencia de movidas locales para encontrar los trayectos más cortos
- Políticas de decisión estocástica

4.3. CARACTERÍSTICAS DE LAS HORMIGAS ARTIFICIALES

La definición de las hormigas artificiales es de naturaleza doble. Por un lado recogen todos los rasgos encontrados en el comportamiento de las hormigas reales en su búsqueda del camino más corto entre el nido y una fuente de comida. Por el otro, han sido fortalecidas con algunas capacidades que no se encuentran en las hormigas reales, necesarias para la solución de difíciles problemas de optimización. La caracterización de las hormigas artificiales comprende tanto las capacidades inspiradas por la naturaleza como aquellas concebidas por el campo de la inteligencia artificial para su reforzamiento.

El siguiente grupo de características corresponden a todas las capacidades que han sido inspiradas por las capacidades análogas encontradas en las hormigas reales [14, 22].

Cooperación entre las hormigas artificiales: Al igual que las colonias de hormigas reales, los algoritmos hormigas están compuestos por una población de agentes conocidos como hormigas artificiales que cooperan globalmente para encontrar una buena solución al problema bajo consideración. Cada hormiga artificial puede por si sola encontrar una solución factible, pero las soluciones de alta calidad son el resultado de la cooperación entre las hormigas de toda la población.

Stigmergy y el rastro de feromona: Las hormigas reales modifican su medio ambiente depositando feromona en cada uno de los trayectos que ellas visitan para

evidenciar la preferencia de un trayecto sobre otro. Las hormigas artificiales cambian la información numérica que esta localmente almacenada en el estado del problema en el que ellas están. En los algoritmos hormigas el *rastro de feromona artificial* hace las veces del rastro de feromona dejado por las hormigas reales. El rastro de feromona artificial se comporta como un tipo de información numérica distribuida que es modificada por las hormigas para reflejar la experiencia acumulada en la búsqueda de la solución. Esta forma de comunicación indirecta conocida como stigmergy es la que hace posible el conocimiento y aprendizaje colectivo de la población de hormigas artificiales. La evaporación del feromona en los algoritmos hormigas es usado de manera análoga a la empleada por la colonia de hormigas reales, para de olvidar el pasado y de esa forma dirigir la búsqueda hacia nuevas direcciones sin ser restringida por las decisiones tomadas anteriormente.

Búsqueda de la solución con mínimo costo y movidas locales: Las hormigas reales deben encontrar el camino más corto entre el nido y la fuente de comida, las hormigas artificiales deben hallar la solución con el mínimo costo al problema específico que esta siendo resuelto. Las hormigas reales realizan la búsqueda caminando a través de trayectos adyacentes del terreno, mientras que las hormigas artificiales la realizan moviéndose paso a paso a través de los estados adyacentes (nodos en el grafo G) del problema.

Políticas de transición estocásticas: Tanto las hormigas reales como las artificiales aplican políticas de decisión probabilísticas para moverse a través de los estados adyacentes. Las probabilidades de transición son una función de dos aspectos: información a priori representada por las especificaciones del problema bajo consideración (equivalente a la estructura del terreno en el que se mueven las hormigas reales), y modificaciones en el rastro de feromona artificial inducido por la experiencia de la búsqueda.

De acuerdo a Di Caro, Dorigo y Gambardella [1999], las características de las hormigas artificiales que corresponden a todas aquellas capacidades que no tienen su contraparte en las hormigas reales son las siguientes:

- Se mueven en un *mundo discreto* y sus movidas consisten en transiciones entre estados discretos.
- Usan un tipo de *memoria limitada* que almacena los arcos que han sido visitados y los costos asociados a ellos.
- Utilizan la *evaluación de los costos de las soluciones* que construyen para determinar la cantidad de feromona que depositaran para actualizar los rastros de feromona de los arcos que visitaron. De esta forma, la actualización de los rastros de feromona son una función de la cualidad de la solución generada, que ayuda a dirigir la búsqueda de futuras hormigas hacia soluciones de alta calidad.
- La forma en que se actualización los rastros de feromonas es específico del problema y casi siempre no refleja el comportamiento de las hormigas reales. En muchos casos como el de los algoritmos ACO, las hormigas artificiales actualizan el rastro de feromona sólo después de haber generado una solución y no durante la construcción de la misma.
- En las colonias de hormigas reales la intensidad del feromona en los trayectos decrece en el tiempo a causa de su evaporación. La evaporación del rastro de feromona debería ayudar a las hormigas a dirigir la búsqueda hacia mejores soluciones sin ser restringida por las decisiones pasadas. Sin embargo, los experimentos de doble trayecto demostraron que la evaporación del feromona en las hormigas reales no juega un papel evidente en la obtención del camino más corto entre el nido y la fuente de comida. Por el contrario, para la colonia de hormigas artificiales puede ser un mecanismo muy útil a la hora de encontrar buenas soluciones a un problema. Probablemente, la evaporación del feromona tiene más importancia en las hormigas artificiales debido al

hecho de que los problemas de optimización que enfrentan estas hormigas es mucho más complejo que aquellos que deben resolver las hormigas reales. La *evaporación artificial del feromona* en la colonia de hormigas artificiales es simulada mediante la aplicación de una regla de evaporación definida, que por lo general, consiste en decaer la intensidad del feromona artificial a una tasa constante, durante la actualización de los rastros de feromona. Este mecanismo puede ser visto como una estrategia de diversificación que favorece la exploración de diferentes trayectos durante el proceso de búsqueda. Así mismo tiene la función de limitar el valor máximo que puede alcanzar el rastro de feromona.

4.4. APLICACIÓN DE LOS ALGORITMOS HORMIGAS

El algoritmo *Sistema hormiga* (AS) fue el progenitor de todos los algoritmos hormigas y su importancia reside en que ha sido el prototipo de un gran número de ellos con exitosas aplicaciones [6, 14, 22]. Las primeras aplicaciones del algoritmo fueron hechas usando el problema del agente viajero (TSP). Las principales razones por las cuales se eligió el problema TSP fueron: es uno de los problemas NP-hard más estudiados dentro de los problemas de optimización combinatoria, es un problema de búsqueda del trayecto más corto en el cual la metáfora del comportamiento de la colonia de hormigas era fácil de implementar, y por último, es un problema didáctico (fácil de entender y sin muchas tecnicidades). El problema TSP consiste en encontrar el tour cerrado de longitud mínima que conecte n ciudades (componentes del problema) que deben ser visitadas sólo una vez. Una definición general del problema TSP es la siguiente. Se considera un conjunto de nodos N que representan las ciudades, y un conjunto de arcos E conectados a los nodos. d_{ij} es la longitud del arco $(i, j) \in E$, que es la distancia entre las ciudades i y j. El problema TSP es el problema de encontrar un circuito hamiltoniano de mínima longitud en el

grafo $G = (N, E)$ donde un circuito hamiltoniano en el grafo G es un tour cerrado que visita cada uno de los $n = |N|$ nodos, sólo una vez, y su longitud está dada por la sumatoria de las longitudes de todos los arcos que lo componen.

Se probaron tres instancias del algoritmo AS aplicadas al problema TSP propuestas en [15] y publicadas en reportes técnicos como [8, 10, 11]. Estos algoritmos fueron llamados: *hormiga- densidad* (ant-density), *hormiga-cantidad* (ant-quantity) y *hormiga-ciclo* (ant-cycle), y difieren por la forma como actualizan el rastro de feromona artificial. En los algoritmos hormiga- densidad y hormiga-ciclo las hormigas artificiales actualizan el rastro de feromona durante la construcción de la solución (después de moverse de un nodo a otro), mientras que en el algoritmo hormiga-ciclo las hormigas lo actualizan después de que han construido un tour (solución) completo. Estudios preliminares corridos sobre varias instancias del problema TSP demostraron que el algoritmo hormiga-ciclo se desempeñaba mejor que los otros [14]. Como consecuencia, el estudio del algoritmo AS se basó en las características de hormiga-ciclo, que más tarde fue llamado simplemente Sistema hormiga (AS); el resto de los algoritmos fueron abandonados. El algoritmo que se presenta a continuación es el algoritmo AS definido por las características de hormiga-ciclo.

4.4.1. Sistema hormiga (AS)

En el algoritmo AS las hormigas artificiales construyen soluciones (tours) del problema TSP moviéndose de una ciudad a otra en el grafo. El algoritmo ejecuta un número máximo de $t_{\max}$ iteraciones representadas por la letra t.

Construcción de las soluciones: Durante cada iteración m hormigas construyen en paralelo una solución que empieza vacía, ejecutando n pasos (cada uno asociado a

una ciudad-componente) en el que una regla de decisión probabilística (regla de transición) es aplicada. Esto significa que si la hormiga $k-ésima$ estando en el nodo i elige moverse al nodo j de acuerdo a la regla, el arco (i,j) es adherido a la solución bajo construcción.

Regla probabilística de transición: Es conocida como la *regla proporcional aleatoria* (random proporcional rule). La probabilidad con la que la hormiga k elige ir de la ciudad i a la ciudad $j \in N_i^k$ mientras construye su solución en la $t-ésima$ iteración del algoritmo es:

$$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in N_i} [\tau_{il}(t)]^\alpha \cdot [\eta_{il}]^\beta} \qquad \forall j \in N_i^k \qquad (8)$$

Donde:

- La cantidad del rastro de feromona en el arco (i,j) en el tiempo t $\{\tau_{ij}(t)\}$: representa la deseabilidad de escoger la ciudad j estando en la ciudad i.
- La visibilidad del arco (i,j) en el tiempo t $\left\{\eta_{ij} = 1/d_{ij}\right\}$: representa el valor heurístico de moverse de la ciudad i a la ciudad j.
- $\alpha \in (0,1)$ y $\beta \in (0,1)$ son los parámetros que controlan la influencia relativa del rastro de feromona y el valor heurístico respectivamente. Si $\alpha = 0$, las ciudades más cercanas tienen más probabilidad de ser seleccionadas. Si por el contrario $\beta = 0$, la elección estará determinada únicamente por el nivel de feromona, dirigiendo al algoritmo a una situación de estancamiento que lleva a la construcción de soluciones de muy baja calidad.
- $N_i^k \subseteq N_i$ es el conjunto de nodos en la vecindad del nodo i que la hormiga k no ha visitado aún. Los nodos N_i^k son seleccionados de N_i usando la

memoria asociada a cada hormiga M^k. La memoria contiene la lista de las ciudades ya visitadas por la hormiga, llamada *lista tabú* [8, 10, 11]. Es usada para definir el grupo de ciudades que la hormiga k ubicada en la ciudad i aún no ha visitado.

Después de que todas las hormigas han completado su solución, se aplica la regla de evaporación del feromona en todos los arcos (su papel principal es impedir el estancamiento del algoritmo) y cada hormiga k deposita una cantidad de feromona $\Delta\tau_{ij}^k(t)$ en los arcos que haya usado (reforzamiento de los rastros de feromona):

$$\Delta\tau_{ij}^k(t) = \begin{cases} \dfrac{1}{L^k(t)} & \text{Si el arco } (i,j) \in T^k(t) \\ 0 & \text{Si el arco } (i,j) \notin T^k(t) \end{cases} \qquad (9)$$

Donde:

- $T^k(t)$: es el tour (solución) hecho por la hormiga k en la iteración t.
- $L^k(t)$: es la longitud del tour hecho por la hormiga k en la iteración t.

Actualización de los rastros de feromona: La cantidad de feromona adherido por las hormigas y la evaporación de feromona, son implementados la regla (10) aplicada a todos los arcos:

$$\tau_{ij}(t+1) = (1-\rho)\cdot\tau_{ij}(t) + \Delta\tau_{ij}(t) \qquad (10)$$

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^k(t) \qquad (11)$$

Donde:

- m: es el número de hormigas de la población (parámetro del algoritmo).

- $\rho \in (0,1]$ y $(1-\rho)$: es el coeficiente de decrecimiento del rastro de feromona.

La cantidad inicial del rastro de feromona, es decir en la iteración $t = 0$, se hace igual a un valor constante y pequeño $\tau_{ij}(0) = \tau_0$ para todos los arcos.

De acuerdo a Colorni, Dorigo y Maniezzo [1996], el algoritmo AS aplicado al problema TSP podía ser utilizado para resolver otros problemas de optimización combinatoria como el problema QAP y el JSP adaptándolos apropiadamente. Los requerimientos para poder aplicar el algoritmo AS a problemas de optimización combinatoria eran los siguientes:

- Una representación apropiada del grafo del problema sobre el cual las hormigas artificiales pudieran construir las soluciones
- Un mecanismo autocatalítico de retroalimentación
- Una heurística que defina la construcción de las soluciones
- Un método de satisfacción de restricciones (lista tabú)

De ellos, los más difíciles de afrontar a la hora de obtener la versión adaptada AS-XXX (donde XXX representa el tipo de problema) del algoritmo AS al problema específico son: encontrar la representación apropiada del grafo y determinar la heurística de construcción.

4.4.2. AS aplicado al problema QAP (AS-QAP)

QAP fue el segundo problema después del TSP, al que se aplico el algoritmo AS para resolverlo. El motivo fue su interpretación como una generalización del problema TSP[14] (Di Caro, Dorigo y Gambardella [1999]). La versión adaptada del algoritmo

[14] El problema TSP puede ser visto como el problema de asignar a cada una de las n ciudades un número aleatorio diferente entre 1 y n (Di Caro, Dorigo y Gambardella [1999]).

AS al problema QAP, conocido como AS-QAP [9], emplea el algoritmo AS usando una heurística max-min específica para QAP que permitía calcular los valores de η. El algoritmo AS-QAP se probó en un grupo de instancias del problema QAP y las soluciones que arrojó demostraron ser de la misma calidad de las soluciones obtenidas por metaheurísticas como SA y GA (Di Caro, Dorigo y Gambardella [1999]).

4.4.2.1. Representación del grafo para el problema QAP. La característica clave en la representación del grafo es la identificación de los tipos de componentes del problema y las interrelaciones entre ellos (Montgomery [2002]). De la estructura de esta representación depende la interpretación del rastro de feromona artificial en el problema específico. Las hormigas artificiales construyen las soluciones recorriendo un grafo de construcción $G = (N, E)$ (del que ya se ha venido hablando) en el que los componentes del problema conforman los vértices o nodos N del grafo y las interrelaciones o conexiones entre ellos están dadas por los arcos E. Problemas como el TSP sólo tiene un único tipo de componente identificado como *ciudades*, cada componente de este tipo corresponde a una ciudad. Sin embargo, en problemas como el QAP el conjunto de componentes N consiste de dos o más subconjuntos de componentes (tipos de componentes) $N = N_1 \cup N_2 \cup ... \cup N_n$. El problema QAP tiene dos tipos de componentes: N_i : *localidades* y N_r : *actividades*, y las relaciones o conexiones se dan entre ellos (localidades-actividades) mas no dentro de ellos. De esta forma, los elementos que hacen parte de la solución de un problema QAP tienen la forma $(i,r) \in N_i \times N_r$, donde $N = N_i \cup N_r$, $N_i \cap N_r \neq \phi$ y los índices i y r representan a una localidad y a una actividad del problema respectivamente.

4.4.2.2. Descripción del algoritmo AS-QAP. El problema QAP de orden n consiste en encontrar la asignación de n actividades a n localidades con el mínimo

costo. En el algoritmo AS-QAP la asignación de las actividades a las localidades se hace en un orden fijo predeterminado.

Las diferencias entre AS-QAP y el algoritmo AS aplicado al problema TSP radican en la interpretación que se le da al rastro de feromona τ_{ir}, y a los valores heurísticos η definidos por la heurística determinada específicamente para el problema QAP.

Construcción de las soluciones: A cada paso n (correspondiente a la actividad a ser asignada de acuerdo al orden prefijado) una actividad r es asignada probabilísticamente a alguna localidad i. Entonces, el arco (elemento de la solución) (i,r) es adherido a la solución parcial.

Regla probabilística de transición: La probabilidad de que la hormiga k asigne la actividad r a la localidad $i \in N_1^k$ mientras construye su solución en la $t-ésima$ iteración del algoritmo es:

$$p_{ir}^{k}(t) = \frac{[\tau_{ir}(t)]^{\alpha} \cdot [\eta_{ir}]^{\beta}}{\sum_{l \in N_1} [\tau_{lr}(t)]^{\alpha} \cdot [\eta_{lr}]^{\beta}} \qquad \forall i \in N_i^k \qquad (12)$$

Donde:

- La cantidad del rastro de feromona en el arco (i,r) en el tiempo t $\{\tau_{ir}(t)\}$: representa la deseabilidad de asignar la actividad r a la localidad i.
- La visibilidad del arco (i,j) en el tiempo t $\left\{\eta_{ir} = 1/e_{ir}\right\}$: representa el valor heurístico de asignar la actividad r a la localidad i. El valor e_{ir} es el valor definido por la heurística max-min para el cálculo del valor de η_{ir} en el problema QAP.

- $N_1^k \subseteq N_1$ es el conjunto de localidades a los que la hormiga k no les ha asignado aún actividad. Los componentes N_i^k son seleccionados de N_i usando la memoria asociada a cada hormiga M^k. La memoria o lista tabú, contiene la lista de las localidades a las que la hormiga k ya les ha asignado actividad. Es usada para definir el grupo de localidades a los que la hormiga k le falta por asignar actividad.

La cantidad de feromona $\Delta\tau_{ij}^k(t)$ que cada hormiga k deposita en los arcos que usó para la construcción de su solución esta dada por (13):

$$\Delta\tau_{ir}^k(t) = \begin{cases} \dfrac{1}{L^k(t)} & \text{Si el arco } (i,r) \in T^k(t) \\ 0 & \text{Si el arco } (i,r) \notin T^k(t) \end{cases} \qquad (13)$$

Donde:

- $T^k(t)$: es la asignación generada por la hormiga k en la iteración t.
- $L^k(t)$: es el costo de la asignación generada por la hormiga k en la iteración t.

Actualización de los rastros de feromona: La cantidad de feromona adherido por las hormigas y la evaporación de feromona, son implementados por la regla (14) aplicada a todos los arcos:

$$\tau_{ir}(t+1) = (1-\rho)\cdot\tau_{ir}(t) + \Delta\tau_{ir}(t) \qquad (14)$$

$$\Delta\tau_{ir}(t) = \sum_{k=1}^{m} \Delta\tau_{ir}^k(t) \qquad (15)$$

Los parámetros α, β, ρ y m tienen la misma connotación y juegan el misma papel que en el problema TSP.

4.4.3. Extensiones y mejoras del algoritmo AS

Los resultados de las aplicaciones del algoritmo AS al problema TSP aunque eran buenos no podían competir con los de metaheurísticas ya establecidas para resolverlo. Esta situación motivo a un grupo de investigadores a desarrollar extensiones y mejoras de las ideas básicas de AS para que tuviera un mejor desempeño. Estas mejoras se convirtieron posteriormente en las acciones *daemon*[15] utilizadas con más frecuencia por los algoritmos hormigas que siguieron al algoritmo AS. A continuación se resumen las acciones daemon implementadas al algoritmo AS para mejorar y/o generar extensiones del mismo que evolucionaron en otro tipo de algoritmo hormiga.

Estrategia elitista: Fue la primera mejora introducida al algoritmo AS [10]. El término reside en la analogía de su uso con la estrategia elitista empleada en los algoritmos genéticos. Consiste en reforzar con mayor cantidad de feromona a aquellos rastros de feromona correspondientes a los arcos que pertenecen a la mejor solución encontrada por el algoritmo hasta el momento (hasta la $t-ésima$ iteración) denominada *mejor solución global*. Esto significa que en cada iteración, cuando se lleve a cabo la actualización de los rastros de feromona, los rastros de feromona asociados a los arcos de la mejor solución global $T^{gb}(t)$ adquieren una cantidad adicional de feromona. La cantidad de feromona adicional $\Delta\tau_{ij}^{gb}(t)$ que reciben estos arcos en la $t-ésima$ iteración esta dada por la ecuación (16):

[15] Son mecanismos opcionales no propios de la metaheurística, pero que pueden ser empleados por los algoritmos para optimizarlos. Entre ellos se cuentan las hibridaciones.

$$\Delta\tau_{ij}^{gb}(t)=\begin{cases}\dfrac{e}{L^{gb}(t)} & \text{Si el arco } (i,j)\in T^{gb}(t)\\ 0 & \text{Si el arco } (i,j)\notin T^{gb}(t)\end{cases} \qquad (16)$$

Los arcos pertenecientes a $T^{gb}(t)$ son reforzados con la cantidad adicional de feromona $\dfrac{e}{L^{gb}(t)}$, donde e es un entero positivo y $L^{gb}(t)$ es el valor de la función objetivo de $T^{gb}(t)$.

Regla proporcional pseudo-aleatoria: Corresponde a una mejora en la regla probabilística de transición. Esta nueva regla de transición es utilizada por primera vez en el algoritmo ACS [16], un algoritmo que supera en desempeño al algoritmo AS incrementando la importancia de la experiencia adquirida por las hormigas con respecto a la exploración del espacio de búsqueda a través de dos mecanismos [22]. Uno de ellos es la regla de transición *proporcional pseudo-aleatoria* (pseudo-random proporcional rule) y el otro es la estrategia elitista usada en la actualización de los rastros de feromona. La nueva regla de transición usada por la hormiga k para adherir un elemento a la solución parcial en la iteración t, tiene doble función:

Explotación: La regla de decisión explota el conocimiento disponible acerca del problema, esto es, la información heurística; y el conocimiento aprendido durante el proceso de búsqueda en la forma de rastro de feromona. q es una variable aleatoria uniformemente distribuida entre [0,1], y $q_0 \in [0,1]$ es un parámetro ajustable. La explotación se cumple para el siguiente valor de q.

$$\text{Si } q \le q_0 :\ p_{ij}^{k}(t)=\begin{cases}1 & \text{Si } j = \arg\max_{j\in N_i^k} \left[\tau_{ij}(t)\right]\cdot\left[\eta_{ij}\right]^{\beta}\\ 0 & \text{Si } j \neq \arg\max_{j\in N_i^k} \left[\tau_{ij}(t)\right]\cdot\left[\eta_{ij}\right]^{\beta}\end{cases} \qquad (17)$$

Exploración: La regla opera como un mecanismo de exploración parcial equivalente a la regla de transición original del algoritmo AS. La exploración se cumple para el siguiente valor de q.

$$\text{Si } q > q_o: \qquad p_{ij}^{k}(t) = \frac{\left[\tau_{ij}(t)\right]^{\alpha} \cdot \left[\eta_{ij}\right]^{\beta}}{\sum_{l \in N_i} \left[\tau_{il}(t)\right]^{\alpha} \cdot \left[\eta_{il}\right]^{\beta}} \qquad \forall j \in N_i^k \qquad (18)$$

El parámetro q_0 controla el grado de exploración y determina si el proceso de búsqueda se debe concentrar en las mejores soluciones encontradas o si elige examinar regiones del espacio de búsqueda sin explorar. En esta media, la función de explotación funciona como una estrategia de intensificación y la de exploración como una estrategia de diversificación.

La estrategia elitista empleada en el algoritmo ACS da origen a una nueva forma de actualización del rastro de feromona conocida como *actualización fuera de línea*: al final de una iteración, una vez que todas las hormigas han construido su solución, la cantidad de feromona adicional Δ_{ij}^{mejor} es adherida a los arcos usados por la hormiga que encontró la mejor solución desde el principio de la búsqueda. La regla de actualización fuera de línea del rastro de feromona esta dada por (19):

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \rho \cdot \Delta_{ij}^{mejor}(t) \qquad (19)$$

La *mejor* hormiga (solución) puede ser: *mejor-iteración*, es decir, la mejor solución de la iteración, o *mejor-global*, la mejor solución global.

Los algoritmos hormigas que partieron del algoritmo AS y superaron su desempeño, además del algoritmo ACS, son AS_{rank}, *Hormiga-Q aprendizaje* (Ant-Q learning), *MAX-MIN Sistema hormiga* (MMAS) [26, 27] y *ACS-3-opt* aplicados en un principio a problemas TSP. De ellos sólo MMAS tiene una versión adaptada al problema QAP

conocido como MMAS-QAP. Los algoritmos hormigas diseñados originariamente para resolver problemas QAP fueron AS-QAP^b [12] (una versión adaptada de AS-QAP), *Hormigas-QAP* (ANTS-QAP) [28] y *Sistema de hormigas híbrido-QAP* (HAS-QAP) [18]. Los algoritmos MMAS-QAP y HAS-QAP fueron comparados con algunas de las mejores heurísticas disponibles para el problema QAP en dos clases de instancias: las aleatorias-no estructuradas (tipos (a) y (b)) y las estructuradas (tipos (c) y (d)). Ambos algoritmos se encuentran dentro de los de mejor desempeño resolviendo instancias estructuradas (Di Caro, Dorigo y Gambardella [1999]).

Las aplicaciones y extensiones del algoritmo AS, fueron posteriormente unificadas en un meta-esquema conocido como la metaheurística *Optimización por colonia de hormigas* (ACO), que describe el comportamiento generalizado de una colonia de hormigas artificiales en la búsqueda de soluciones a un problema de optimización (Bonabeau, Dorigo y Theraulaz [2000]).

Capítulo V

LA METAHEURÍSTICA ACO

La metaheurística ACO es un sistema definido a posteriori que no formó parte de los estudios originales de los algoritmos hormigas. Fue debido al éxito de los algoritmos hormigas creados hasta el momento, que Di Caro y Dorigo [1999] después de un análisis cuidadoso de las características de un número de ellos (la mayoría aplicaciones y extensiones inspiradas por el algoritmo AS) en un esfuerzo de sintetización, propusieron el meta-esquema ACO con el fin de proveer un sistema generalizado que reuniera una clase de instancias de los algoritmos hormigas con características similares dentro de un mismo campo; que al mismo tiempo sirviera de marco de referencia para el diseño de nuevas instancias (algoritmos) ACO. A todos los algoritmos hormigas que representan instancias de la metaheurística ACO se les conoce con el nombre de *algoritmos ACO*. Por esa razón, todos los algoritmos ACO son considerados también algoritmos hormigas, pero lo contrario no siempre se cumple, ya que no todos los algoritmos hormigas pueden ser considerados instancias de la metaheurística ACO, es decir, algoritmos ACO[16].

Los algoritmos ACO son esencialmente algoritmos de construcción, por esa razón, la metaheurística ACO puede ser interpretada como una extensión de las heurísticas de construcción basada en tres aspectos claves que la hacen superior a las técnicas heurísticas (Dorigo y Stützle [2000]).

Componente estocástico (probabilístico): Este componente lleva a los algoritmos ACO a construir una gran variedad de soluciones diferentes y por ende, a explorar

[16] El término algoritmo ACO será usado para referirse a cualquier instancia específica de la metaheurística ACO. Mientras que los algoritmos hormigas hacen referencia a cualquier algoritmo hormiga que sigue todos los aspectos descritos anteriormente, pero no necesariamente sigue todos los de la metaheurística ACO, como es el caso del algoritmo hormiga HAS.

una mayor cantidad de soluciones de las que exploran las heurísticas de construcción, como la voraz (greedy). Este componente esta representado en las reglas de transición.

Uso de información heurística: Esta siempre disponible para la mayoría de los problemas de optimización o por lo general es fácil de calcular. Este tipo de información puede guiar a las hormigas artificiales hacia las buenas soluciones. Por ejemplo, la heurística max-min definida para el algoritmo AS-QAP proveía un orden de asignación de las actividades que buscaba simular la asignación natural encontrada en las instancias estructuradas del problema QAP (problemas reales) y que ubica actividades con mayor flujo en localidades con la menor distancia posible entre ellas.

Proceso de aprendizaje: La información brindada por los rastros de feromona es usada por los algoritmos ACO para almacenar la experiencia adquirida por las hormigas acerca de la utilidad de los componentes del problema. Es utilizada para la construcción de mejores soluciones en las futuras iteraciones del algoritmo.

5.1. REPRESENTACIÓN GENERALIZADA DE UN PROBLEMA DE OPTIMIZACIÓN

De acuerdo a Dorigo y Stützle [2000], un problema de optimización combinatoria de mínimo (o máximo) costo (S, f, Ω), donde S es el conjunto de posibles soluciones, f es la función objetivo que asigna a cada posible solución $s \in S$ un valor (costo) de la función objetivo $f(s,t)$ en la iteración t, y Ω es un conjunto de restricciones, puede ser representado para que sea explotado por las hormigas artificiales en la búsqueda de la solución óptima global $s_{opt} \in S$, es decir, la solución con el mínimo costo que satisface las restricciones Ω, a través de la siguiente caracterización.

- Un conjunto $P = \{C_1, C_2, C_3, ..., C_n\}$ integrado por un número finito de tipos de componentes, donde el conjunto $C = C_1 \cup C_2 \cup ..., C_{n-1} \cup C_n$ comprende los componentes de todos los tipos.
- Cada conjunto $C_i = \{c_1, c_2, c_3, ..., c_{C_i}\}$ esta compuesto por un número finito de componentes del tipo i.
- Los *estados* del problema son definidos en términos de secuencias $x = \langle c_i, c_j, ..., c_k, ... \rangle$ sobre los elementos de C. X denota el conjunto de todas las posibles secuencias. La longitud de cada secuencia x esta dada por el número de componentes que la componen, y es expresada por $|x|$.
- El conjunto finito de *restricciones* Ω define el conjunto de estados factibles $\tilde{X}$ del problema, con $\tilde{X} \subseteq X$.
- Un conjunto S^* de soluciones factibles, con $S^* \subseteq \tilde{X}$ y $S^* \subseteq S$.
- Un costo $f(s,t)$ asociado a cada posible solución $s \in S$.

Dada la representación del problema, las hormigas artificiales pueden construir las posibles soluciones moviéndose sobre un arco de construcción $G = (C, L, W)$, donde los vértices corresponden a los componentes C, los arcos $L = \{l_{c_i c_j} \mid c_i, c_j \in C\}$ corresponden al conjunto finito de posibles conexiones entre los elementos de C y W es el conjunto de los pesos asociados a los componentes C, a las conexiones L o a ambos. La factibilidad de las soluciones esta definido con respecto al conjunto de restricciones $\Omega(C, L, \theta)$ del problema, asignadas sobre los elementos de C y L, donde θ indica que las restricciones pueden cambiar en el tiempo. En el problema QAP definido en la sección 2.1, el conjunto de componentes $C = C_i \cup C_r$ es la unión de dos tipos de componentes: localidades y actividades, L es el conjunto de arcos correspondientes a las parejas de índices (i,r) formadas por la asignación de una

actividad r a una localidad i, y W es el conjunto de costos asociados a las parejas (i,r).

5.2. COMPORTAMIENTO DE LAS HORMIGAS ARTIFICIALES EN ACO

En los algoritmos ACO como algoritmos de construcción que son, cada hormiga de la colonia de hormigas artificiales simula un procedimiento de construcción estocástica que construye soluciones del problema definido por el grafo de construcción $G=(C,L,W)$ y por las restricciones $\Omega(C,L,\theta)$ (Dorigo y Stützle [2000]). El movimiento de las hormigas en el grafo no es aleatorio, por el contrario, sigue una política de construcción que es función de las restricciones Ω del problema. Son ellas las que determinan el grado de flexibilidad con que se deja que las hormigas artificiales construyan soluciones factibles y que sea posible la construcción de soluciones infactibles. En el grafo G los componentes $c_i \in C$ o las conexiones $l_{ij} \in L$ tienen asociado: a) un rastro de feromona τ: τ_i si esta asociado a los componentes y τ_{ij} si esta asociado a las conexiones. Se desempeña como una memoria de largo plazo acerca de todo el proceso de búsqueda de las hormigas, que ellas mismas actualizan. b) un valor heurístico η: η_i si esta asociado a los componentes y η_{ij} si esta asociado a las conexiones. Representa una información a priori acerca de la instancia del problema que no es proveída por las hormigas como en el caso del rastro de feromona. Las hormigas artificiales construyen soluciones adhiriendo iteratívamente elementos a las soluciones parciales elegidos de acuerdo a una regla probabilística que tiene en cuenta el rastro de feromona artificial y la información heurística encontrada en los componentes o en las conexiones del grafo G del problema. Dorigo y Stützle [200] definen el grupo de propiedades que caracterizan a las hormigas artificiales de los algoritmos ACO.

- Explotan el grafo $G = (C, L, W)$ que representa al problema a ser resuelto, para buscarle soluciones factibles s de mínimo costo. Es decir, soluciones s tales que $\hat{f}_s = \min_s f(s,t)$.
- Tienen una memoria de corto plazo M^k que almacena información acerca de los elementos adheridos hasta el momento. La memoria puede ser usada para tres propósitos: a) construir soluciones factibles, b) evaluar la solución encontrada y c) recordar e identificar los componentes o conexiones visitados por la hormiga, para actualizar los rastros de feromonas asociados.
- Se le puede asignar un *estado inicial* x_s^k y una o más *condiciones de terminación* e^k. Generalmente, el estado inicial es expresado en función de la longitud de la secuencia. De esa forma, el estado inicial estada dado por: a) una secuencia con longitud unitaria (una secuencia con un único elemento), o como una secuencia (una secuencia sin ningún elemento).
- Si estando en el estado $\langle x_{r-1}, i \rangle$ la condición de terminación no es satisfecha todavía. La hormiga k se mueve al vértice j dentro de la vecindad del nodo i N_i^k, es decir, al estado $\langle x_r, j \rangle \in X$. Casi siempre las movidas favorecen a estados factibles, guiados por los valores heurísticos η, definidos apropiadamente para el tipo de problema o a través del uso de la memoria M^k.
- Seleccionan la movida aplicando una regla de decisión probabilística definida en función de: a) rastros de feromona disponibles en la vecindad del nodo i N_i^k correspondiente al estado actual, b) la memora M^k que almacena las movidas realizadas por la hormiga hasta ese estado (estado actual), y c) las restricciones Ω del problema.
- El procedimiento de construcción de la hormiga k para cuando al menos una de las condiciones de terminación e^k es satisfecha.

- La hormiga k puede realizar diversos tipos de actualización del rastro de feromona: a) *actualización en línea paso a paso* (online step-by-step pheromone update): cada vez que la hormiga adhiere un elemento a la solución parcial actualiza los rastros de feromona asociados a los componentes que lo conforman o a las conexiones entre los componentes y b) *actualización en línea retrasado* (online delayed pheromone update): únicamente después de construida la solución se actualizan los rastros de feromonas asociados a los componentes o a las conexiones usadas.

5.3. EL META-ESQUEMA ACO

El meta-esquema ACO es el sistema unificado que generaliza las características en común de los algoritmos ACO que componen la metaheurística ACO y que sirve de marco de referencia para el diseño de nuevas instancias ACO. En esa medida, el meta-esquema ACO representa la estructura básica que caracteriza a todo algoritmo ACO. La estructura definida por el meta-esquema ACO (ver figura 8) comprende la sincronización de tres actividades, resumida en el término *programación de las actividades*. Las tres actividades a programar son: *actividad de las hormigas*, *evaporación del feromona* y *acciones daemon*.

```
Procedimiento Metaheurística ACO
Programación de actividades
    Actividad de las hormigas ( )
    Evaporación del feromona ( )
    Acciones daemon ( )  {opcional}
fin Programación de actividades
fin Procedimiento Metaheurística ACO
```

Figura 8. Estructura de la metaheurística ACO en pseudo-código. El comentario encerrado en llaves indica que el procedimiento Acciones daemon ( ) es opcional

Actividad de las hormigas: Comprende todas las acciones realizadas propiamente por las hormigas artificiales que constituyen los pasos para la construcción de soluciones. Y la posterior evaluación de las soluciones empleada para determinar la cantidad de feromona a adherir sobre los trazos de feromona que corresponden a los componentes o conexiones utilizadas en ellas. El procedimiento de la actividad de las hormigas describe el comportamiento de una colonia de hormigas artificiales que se mueven en paralelo entre estados adyacentes del problema a través de la construcción de trayectos en el grafo G: una trayectoria factible sobre el grafo G es llamada una solución factible s y una trayectoria de mínimo costo es una solución óptima y es indicada por s^*; $f(s,t)$ es el costo de la solución factible y $f(s^*,t)$ es el costo de la solución óptima. Cada hormiga k empieza con una solución parcial inicial $s_k(1)$ compuesta de un elemento, o una solución vacía sin ningún elemento, y adhiere elementos a la solución parcial $s_k(h)$ hasta completar una solución factible s_k, donde h representa el $h-ésimo$ paso de los n pasos de construcción correspondiente a una posible condición de terminación. Los elementos (componentes o conexiones entre componentes) a ser adheridos a $s_k(h)$ son elegidos probabilísticamente dentro de una vecindad apropiadamente definida del último elemento adherido a $s_k(h)$. La decisión estocástica es hecha aplicando una regla de decisión estocástica que hace uso de información local disponible en los vértices visitados (los vértices representan los elementos adheridos que pueden ser componentes de un solo tipo o conexiones entre tipos de componentes): rastro de feromona y valor heurístico. Una vez que la hormiga ha construido una solución, o mientras que la solución esta siendo construida, las hormigas evalúan la solución completa o parcial y adhieren una cantidad de feromona determinada en función de la cualidad de la solución sobre los rastros de feromona asociados a los componentes o conexiones usadas.

Evaporación del feromona: En la actualización del rastro de feromonas las hormigas artificiales en la actividad de las hormigas, sólo contribuyen con la fase de reforzamiento adhiriendo una cantidad de feromona sobre los vértices visitados en la solución que construyen. La fase de evaporación del feromona implementada paralelamente con la de reforzamiento es llevada a cabo aplicando un factor de decrecimiento de la intensidad del feromona. La evaporación del feromona es el procedimiento mediante el cual la intensidad de los rastros de feromonas decrece en el tiempo. Es utilizado para impedir la rápida convergencia del algoritmo hacia regiones sub-óptimas (estancamiento), implementando un mecanismo que olvida las "malas" decisiones pasadas y favorece la exploración de nuevas áreas (áreas no exploradas) del espacio de búsqueda.

Acciones daemon: Este procedimiento implementa acciones centralizadas que no pueden ser desarrolladas por las hormigas artificiales [stigmergy, Aplicaciones ACO]. Las acciones daemon comprenden todos los mecanismos opcionales (extras) que usan información global acerca de la experiencia de la búsqueda. Ejemplos de acciones daemon se encuentran: activación de un mecanismo de optimización local de soluciones (aplicación de alguna heurística de búsqueda local), recopilación de información global empleada para decidir si es útil o no depositar una cantidad de feromona adicional para guiar el proceso de búsqueda desde una perspectiva no local (no utiliza directamente la información local contenida en los vértices visitados en el grafo). Las *actualizaciones fuera de línea de los rastros de feromona* son actualizaciones del rastro de feromona llevadas a cabo por acciones daemon del último tipo. Este mecanismo observa y evalúa las trayectorias (soluciones) encontradas en el grafo G por cada hormiga k de la colonia y sólo deposita una cantidad de feromona extra sobre los componentes o conexiones entre componentes, usados por la hormiga que construyó la mejor solución. El concepto de *daemon* fue introducido por primera vez en la definición de la metaheurística ACO para

caracterizar a aquellos mecanismos en los que sus agentes trabajan como supervisores que con un conocimiento específico del problema, ayudan a las hormigas artificiales a elegir las buenas soluciones o a no desviarlas hacia soluciones sub-optimas.

La planificación de las actividades en la metaheurística ACO es llevada a cabo por el constructor *Programación de actividades*. El constructor no especifica cómo deben ser secuenciadas y sincronizadas las actividades. Por lo tanto, el diseñador es libre de especificar la manera en que estos tres procedimientos interactúan. En general, en los problemas no-distribuidos como los problemas clásicos *NP* – hard (TSP, QAP) se usa normalmente una programación de las actividades estrictamente secuencial, mientras que los problemas distribuidos como el enrutamiento en redes utilizan eficientemente una forma de programación en paralelo fácil de implementar [14, 22]. La interpretación de la metaheurística ACO como marco de referencia para el diseño de nuevas instancias ACO esta determinada por la flexibilidad del constructor para definir la interrelación y comunicación entre los agentes-mecanismos que componen los diferentes procedimientos (actividades).

5.4. ARQUITECTURA MULT-NIVEL (MLA) DE LA METAHEURÍSTICA ACO

La arquitectura multi-nivel definida para la metaheurística ACO asume que en general los algoritmos ACO utilizan una programación adaptativa de la memoria AMP como mecanismo de aprendizaje para dirigir el proceso de búsqueda. AMP implementada en los algoritmos ACO funciona de la siguiente manera (ver Algoritmo 4 pagina 67): después de que todas las hormigas artificiales de la colonia han

construido sus soluciones, se implementa una fase de búsqueda local para optimizar las soluciones (refinarlas); las soluciones optimizadas son evaluadas para actualizar la memoria (rastros de feromona asociados a los elementos que componen las soluciones) y el procedimiento es repetido hasta que se cumple un criterio de terminación.

Algoritmo 4 AMP implementada en un algoritmo ACO

Mientras el criterio de terminación no sea satisfecho

Construir un conjunto de m soluciones $S = \{s_1, s_2, ..., s_m\}$ empleando rastros de feromona y valores heurísticos

Aplicar un algoritmo de búsqueda local a cada solución en S (Acción daemon)

Actualizar los rastros de feromona (memoria de largo plazo)

Fin mientras

Cada uno de los pasos que constituyen el Algoritmo 4 corresponde a las acciones desempeñadas por los agentes en cada uno de los niveles de MLA en la metaheurística ACO. En la figura 9 se observa la estructura MLA de la metaheurística ACO y las interrelaciones entre los diferentes niveles. La interpretación de la estructura MLA de la figura 9 esta dado por la descripción de sus niveles. En el **nivel 0**, los agentes (hormigas artificiales) proveen soluciones iniciales usando un procedimiento de construcción probabilística que tiene en cuenta la información local asociada a los estados (rastros de feromona y valores heurísticos). Los agentes daemon en el **nivel 1** ayudan a las hormigas artificiales a optimizar sus soluciones aplicando un algoritmo de búsqueda local. Y en el **nivel 2** los agentes actualizan los rastros de feromona correspondientes a los elementos usados en las soluciones refinadas. Los agentes de este nivel pueden ser: hormigas artificiales trabajando en la fase de reforzamiento de rastros de feromona, si al algoritmo emplea como tipo de actualización del rastro de feromona, la actualización en línea paso a paso o la actualización en línea retrasada; o agentes daemon, si el tipo de actualización que utiliza el algoritmo es la actualización fuera de línea. Dentro de las

actualizaciones en línea existe otro tipo de agentes además de las hormigas artificiales, encargados del procedimiento de evaporación del feromona. La información actualizada de los rastros de feromona es utilizada de nuevo por las hormigas artificiales para construir soluciones en el nivel 0.

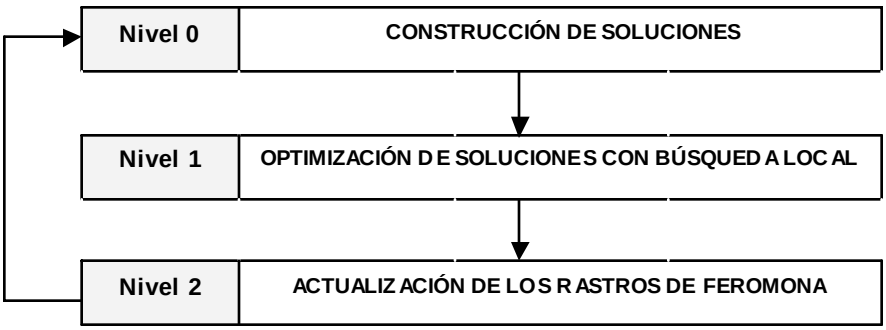


Figura 9. Arquitectura multi-nivel MLA para la versión de la metaheurística ACO con búsqueda local (metaheurística ACO implementando memoria AMP)

La comunicación entre los agentes de cada uno de los niveles se ve reflejada en la figura 9, a través de los conectores que sirven de enlace entre ellos (niveles). Los agentes del nivel 0 interactúan con los del nivel 1, y éstos a su vez lo hacen con los del nivel 2. La conexión del nivel 2 con el nivel 0 indica que los algoritmos ACO son auto-adaptativos, esto significa que desarrollan un proceso de aprendizaje basado en la experiencia adquirida por la recopilación de información brindada por los rastros de feromona sobre la utilidad de los elementos del problema. De esta forma, las hormigas artificiales en el nivel 0 construirán soluciones promisorias que usan preferiblemente elementos que durante el proceso de búsqueda (iteraciones pasadas) han sido contenidas en las buenas soluciones (óptimos locales). Los agentes daemon en el nivel 1 toman estas soluciones y las optimizan aplicándoles un algoritmo de búsqueda local. En el nivel 2 los agentes observan y evalúan las soluciones ya optimizadas para identificar los elementos que hacen parte de ellas y actualizar sus rastros de feromona asociados con ello se les da mayor probabilidad a elementos que

hacen parte de las buenas soluciones de ser elegidos en las iteraciones siguiente por las hormigas artificiales para la construcción de mejores soluciones en el nivel 0. Esto es posible a través del enlace de retroalimentación entre el nivel 2 y el nivel 0 que al repetir el ciclo desarrolla en el algoritmo ACO un proceso de aprendizaje (AMP simula este proceso en los algoritmos ACO que utilizan búsqueda local) acerca de la búsqueda.

La algoritmos ACO que utilizan entre sus acciones daemon la búsqueda local, son en sí mismos métodos híbridos que combinan una heurística de construcción probabilística adaptativa[17] con una heurística de búsqueda local. La razón para utilizar esta opción daemon es porque por lo general las soluciones generadas por los algoritmos de construcción no son tan de buenas como las generadas por los algoritmos de búsqueda local y pueden ser refinadas por estos últimos. Aún así, las heurísticas de búsqueda local por sí solas no generan soluciones de alta calidad. Con base a esta observación, los algoritmos ACO con mejor desempeño para muchos de los problemas de optimización combinatoria *NP* – hard son algoritmos híbridos, de hecho las soluciones de alta calidad para este tipo de problemas son encontradas por algoritmos híbridos que combinan alguna forma de manipulación de las soluciones[18] con la subsiguiente búsqueda local. Se ha demostrado por ejemplo, que para la mayoría de problemas de optimización combinatoria, algoritmos de búsqueda local aplicados sobre soluciones iniciales generadas aleatoriamente producen soluciones alejadas considerablemente de las soluciones óptimas. Las soluciones de alta calidad son encontradas por los algoritmos híbridos ACO explotando las heurísticas de construcción y las de búsqueda local de la siguiente forma: en primer lugar se usa la heurística de construcción probabilística adaptativa con el fin de identificar los

[17] La regla de transición empleada en la construcción de las soluciones utiliza la información local o memoria en forma de rastros de feromona actualizados en cada iteración, con base a la experiencia de la búsqueda.

[18] Se refiere al tratamiento dado a un conjunto de soluciones completas a través de la aplicación de procedimientos o algoritmos específicos. El algoritmo HAS-QAP por ejemplo, maneja dos tipos de manipulación de soluciones: modificación con intercambio de elementos y optimización con búsqueda local.

atributos de las soluciones (utilidad de los elementos que las conforman) de acuerdo a la experiencia pasada reflejada en los rastros de feromona. Con los atributos identificados se guía la heurística de búsqueda para dirigirse hacia regiones promisorias del espacio de búsqueda. La combinación del algoritmo de construcción probabilística adaptativa con el de búsqueda local constituye la programación adaptativa de la memoria (AMP) en los algoritmos ACO híbridos.

5.5. ALGORITMOS ACO APLICADOS AL PROBLEMA QAP

Las aplicaciones actuales de los algoritmos ACO caen dentro de dos importantes clases de problemas de optimización combinatoria [14, 22]: estáticos y dinámicos. Los estáticos son aquellos en los que las características del problema están dadas inicialmente y no cambian mientras que éste es resuelto. En los dinámicos, por el contrario, el problema cambia en el tiempo de corrida y el algoritmo debe ser capaz de adaptarse al nuevo escenario que brindan las características modificadas. El problema QAP se encuentra dentro de la primera clase al igual que el TSP entre otros problemas clásicos *NP* – hard. La tabla 4 contiene el listado de la implementación de los algoritmos ACO al problema QAP clasificados en orden cronológico.

Autores	Año	Referencias	Algoritmo
Colorni, Dorigo y Maniezzo,	1994	[9]	AS-QAP
Dorigo, Gambardella y Taillard	1997	[18]	HAS-QAP
Gambardella y Taillard	1997	[25]	FANT-QAP
Hoos y Stützle	1998	[26, 27]	MMAS-QAP
Colorni y Maniezzo	1998	[28]	AS-QAP[b]
Maniezzo	1998	[12]	ANTS-QAP

Tabla 4. Listado de aplicaciones de algoritmos ACO aplicados al problema QAP clasificados por orden cronológico de aparición

Los algoritmos ACO aplicados al problema QAP son en su totalidad algoritmos híbridos que combinan el componente de construcción probabilística con el de optimización de soluciones con algoritmos de búsqueda local. Además, implementan un constructor *programación de actividades* de la metaheurística ACO secuencial y en orden específico: primero construyen probabilísticamente las soluciones, luego las optimizan y por último actualizan los rastros de feromona, es decir, utilizan una memoria de largo plazo AMP.

Capítulo VI

ALGORITMO HAS-QAP

El algoritmo Sistema hormiga híbrido-QAP (HAS-QAP) es un algoritmo hormiga propuesto originalmente en [18] para resolver el problema QAP. HAS-QAP es un método híbrido que combina el algoritmo ACO AS con una heurística de búsqueda local. Sin embargo, es la única aplicación de los algoritmos AS a problemas QAP y de sus correspondientes mejoras y extensiones que no cae dentro de la metaheurística ACO. A diferencia de los algoritmos ACO, no es un algoritmo de construcción, él reemplaza el componente constructivo de soluciones por uno de modificación de soluciones. De esa forma, el algoritmo empieza con soluciones iniciales completas (a diferencia de las parciales o vacías con las que empiezan los algoritmos ACO) que luego modifica teniendo en cuenta los rastros de feromona. La heurística de búsqueda local es usada después en la optimización de las soluciones modificadas, que más tarde son evaluadas para actualizar los rastros de feromona.

Característica	AS-QAP	ANTS-QAP	MMAS-QAP	FANT-QAP	HAS-QAP
Información heurística	√	√	¬	¬	¬
Población	√	√	√	¬	√
Mecanismo de evaporación del feromona	√	¬	√	¬	√
Heurística de construcción de soluciones	√	√	√	√	¬ [19]

Tabla 5. Caracterización de los algoritmos hormigas aplicados a problemas QAP con base a los aspectos básicos manejados por la metaheurística ACO. Los símbolos √ y ¬ indican la presencia y la no presencia de la característica en el algoritmo respectivamente

La razón principal por la que el algoritmo HAS-QAP no es considerado un algoritmo

[19] La no presencia de esta característica es el motivo principal por el cual el algoritmo HAS-QAP no es considerado un algoritmo ACO.

ACO se ve evidenciado en la tabla 5, que caracteriza a los algoritmos hormigas aplicados a problemas QAP de acuerdo a las ideas básicas que maneja la metaheurística ACO. Entre los aspectos que tienen en común los algoritmos están el uso de un algoritmo de búsqueda local para refinar las soluciones (una acción daemon que funciona como método de hibridización) y el hecho de que los rastros de feromona estén asociados a parejas (i,r) (elementos de las soluciones en problemas QAP) formadas entre localidades y actividades. La tabla 5 resume las diferencias generales encontradas entre los algoritmos basada en la presencia de propiedades ACO como: información heurística, población, mecanismo de evaporación del feromona y heurística de construcción de soluciones.

6.1. DESCRIPCIÓN DEL ALGORITMO HAS-QAP

Al inicio de una iteración t en el algoritmo HAS-QAP (ver Algoritmo 6 página 83), cada hormiga artificial k está asociada a una solución completa π^k del problema definida al final de la iteración anterior (excepto por la primera iteración que parte de soluciones completas generadas aleatoriamente). Cada solución es primero modificada implementando R intercambios teniendo en cuenta la información local brindada por el rastro de feromona, y luego es optimizada usando el algoritmo de búsqueda local *fast descent procedure*. El mecanismo de *modificación* probabilística reemplaza de este modo al mecanismo de construcción probabilística de los algoritmos ACO donde las soluciones no están explícitamente relacionadas con las hormigas artificiales. El problema en HAS-QAP está en la elección de la solución completa con la que partirá cada hormiga artificial k al inicio de una iteración. Para ello han sido definidos los mecanismos de *intensificación* y *diversificación*. Cuando el mecanismo de intensificación esta activado, la solución con la que empieza la hormiga artificial k en la siguiente iteración $t+1$ será la mejor entre la solución inicial y la solución final asociada a ella en t. En el caso en que el mecanismo no

este activado la solución asignada a la hormiga artificial k para empezar la iteración $t+1$ será sencillamente la solución final asociada a ella en t. La actualización de los rastros de feromona se hace con base a la mejor solución global, así que al final de cada iteración se compara la mejor solución obtenida con la mejor solución global, si es mejor, la reemplaza, de lo contrario el algoritmo continúa con la misma. Si después de un número de iteraciones el algoritmo no logra mejorar la mejor solución global, se activa el mecanismo de diversificación implementando un reinicio parcial del algoritmo que consiste en la reinicialización de las soluciones asociadas a las hormigas artificiales con permutaciones generadas aleatoriamente, y consecuentemente la de los rastros de feromona.

A continuación se presentan en detalle los componentes del algoritmo HAS-QAP de acuerdo al diagrama de flujo de los Anexos A y B.

Inicialización de soluciones: Este componente se utiliza en la primera iteración del algoritmo y cada vez que se activa la fase de diversificación en la que se reinicializa el algoritmo. Consiste en la generación aleatoria de m permutaciones $\pi^k(1)$, cada una asociada a una hormiga artificial k. Las permutaciones son posteriormente optimizadas empleando el algoritmo de búsqueda local *fast descent procedure* (explicado más adelante). Estas soluciones corresponden a las soluciones iniciales completas con las que parte el componente de modificación únicamente en la primera iteración del algoritmo y en las iteraciones asociadas a las reinicializaciones; para el resto de las iteraciones las soluciones iniciales completas corresponden a las soluciones asociadas a cada hormiga k al final de la iteración anterior.

Inicialización de los rastros de feromona: Los rastros de feromona en la primera iteración $\tau_{ir}(0)$ son colocados al mismo valor τ_0. Dorigo, Gambardella y Taillard [1997] eligieron hacer que el valor de τ_0 dependiera del valor de una solución

obtenida. Tanto en la primera iteración como en las reinicializaciones esta solución corresponde a π^* la mejor solución entre las permutaciones aleatorias optimizadas con búsqueda local. El valor inicial de los rastros de feromona esta dado entonces por $\tau_0 = \frac{1}{Q \cdot f(\pi^*)}$, donde Q es un parámetro. En las iteraciones mencionadas, π^* corresponde también a la mejor solución global, ya que es la única solución obtenida hasta el momento por el algoritmo contra la que puede compararse la mejor solución obtenida al final de cada una de ellas y determinar de esa forma si hubo o no mejora. En las reinicializaciones el algoritmo parte de cero y nuevamente se deben generar permutaciones aleatorias refinadas después con búsqueda local, y a partir de ellas calcular el nuevo valor de τ_0 con el que se actualizan los rastros de feromona.

Manipulación de las soluciones: Modificación de las soluciones: La modificación es aplicada por cada hormiga artificial a su propia solución asociada π^k al principio de cada iteración, realizando un número de R intercambios sobre ella (ver figura 10). Un intercambio consiste en seleccionar dos posiciones (localidades) i y j del vector que representa a la solución π^k para luego intercambiar sus componentes (actividades) π_i y π_j asociados. La primera posición i es seleccionada aleatoriamente entre 1 y n. La segunda posición $i \neq j$ es elegida de acuerdo a una política de transición que hace uso sólamente de la información local brindada por el rastro de feromona. La política de transición empleada es una regla proporcional pseudo-aleatoria definida por primera vez en el algoritmo ACS descrito en la sección 4.4.3. La regla tiene las mismas dos funciones de ACS, explotación y exploración, pero la fórmulas utilizadas son específicas del algoritmo HAS-QAP. La posición s se elige de acuerdo a los siguientes valores de q.

Explotación: Si $q \leq q_0$:

$$\text{si } j = \arg\max_{j \in N_i^k} \; \tau^k_{i\pi_j}(t) + \tau^k_{j\pi_i}(t)$$

$$\text{si } j \neq \arg\max_{j \in N_i^k} \; \tau^k_{i\pi_j}(t) + \tau^k_{j\pi_i}(t)$$

$$p_{ij}^{k}(t) = \begin{cases} 1 \\ 0 \end{cases} \qquad (20)$$

Donde:

- $p_{ij}^{k}(t)$: corresponde a la probabilidad de que la hormiga k después de haber elegido aleatoriamente la posición i seleccione como segunda posición, a la posición j en la iteración t.
- La cantidad del rastro de feromona en la pareja (i, π_j) en el tiempo t $\tau_{i\pi_j}^{k}(t)$: representa la deseabilidad de asignar la actividad π_j (asociada a la posición j del vector que representa a π^k) a la posición i.
- La cantidad del rastro de feromona en la pareja (j, π_i) en el tiempo t $\tau_{j\pi_i}^{k}(t)$: representa la deseabilidad de asignar la actividad π_i (asociada a la posición i del vector que representa a π^k) a la posición j.
- N_i^k: corresponde a todas las posibles posiciones que pueden ser elegidas por la hormiga k.

Exploración: Si $q > q_o$:

$$p_{ij}^{k}(t) = \frac{\tau_{i\pi_j}^{k}(t) + \tau_{j\pi_i}^{k}(t)}{\sum_{l \neq i \in N_i^k} \tau_{i\pi_l}^{k}(t) + \tau_{l\pi_i}^{k}(t)} \qquad \forall j \in N_i^k \qquad (21)$$

Las soluciones modificadas $\hat{\pi}^k$ corresponden a las soluciones asociadas a cada hormiga artificial k después de aplicados los R intercambios sobre las soluciones π^k.

Solución π^k

5	12	17	2	13	1	4	6	11	19	16	3	7	18	9	10	14	8	15

Iteración[20]	Solución Actual																		Valor F.Obj
	5	12	17	2	13	1	4	6	11	19	16	3	7	18	9	10	14	8	15	17612548
1	5	12	17	2	13	1	19	6	11	4	16	3	7	18	9	10	14	8	15	17532214
2	5	12	17	2	13	1	19	6	11	4	16	3	14	18	9	10	7	8	15	17678545
3	5	6	17	2	13	1	19	12	11	4	16	3	14	18	9	10	7	8	15	17723459
4	5	6	17	4	13	1	19	12	11	2	16	3	14	18	9	10	7	8	15	17228443
5	5	6	1	4	13	17	19	12	11	2	16	3	14	18	9	10	7	8	15	17346786
6	5	6	1	4	13	17	19	12	11	2	3	16	14	18	9	10	7	8	15	17445678

Solución $\hat{\pi}^k$

5	6	1	4	13	17	19	12	11	2	3	16	14	18	9	10	7	8	15

Figura 10. Funcionamiento del mecanismo de modificación en una solución π^k de la instancia **els19,** que genera la solución $\hat{\pi}^k$. Sobre la solución π^k se realiza el primer intercambio y la iteración 1 recibe la solución generada, el próximo intercambio se hace sobre esta última y produce la solución que recibe la iteración 2; la última iteración (iteración 6) recibe la solución correspondiente a $\hat{\pi}^k$ después de realizados los seis intercambios

Manipulación de soluciones: Optimización de soluciones con búsqueda local: Para la optimización de las soluciones, el algoritmo HAS-QAP implementa dos iteraciones del algoritmo de búsqueda local *fast descent procedure* (ver Algoritmo 5 página 78) un algoritmo de mejora iterativa que utiliza la regla pivote *first-improvement 2-opt* que aplica dos intercambios. El algoritmo fast descent procedure aplicado sobre las soluciones $\hat{\pi}^k$ produce soluciones optimizadas $\tilde{\pi}^k$ asociadas a cada hormiga artificial k. La estructura del vecindario de una solución π en éste algoritmo esta definida por el conjunto de soluciones que pueden ser obtenidas intercambiando dos actividades. La diferencia de la función objetivo obtenida

[20] El número máximo de iteraciones del mecanismo de modificación esta dado por el número R de intercambios definido como parámetro. En [18] proponen que $R = n/3$, por eso en la figura el número de iteraciones es $R = 19/3 = 6{,}33$, pero se aproxima al entero más cercano que es 6.

intercambiando las actividades $r = \pi_i$ y $s = \pi_j$ en una solución π puede ser calculado en un tiempo $O(n)$, usando la ecuación (22).

$$\Delta(\pi, i, j) = (a_{ii} - a_{jj})(b_{\pi_j\pi_j} - b_{\pi_i\pi_i}) + (a_{ij} - a_{ji})(b_{\pi_j\pi_i} - b_{\pi_i\pi_j}) + \sum_{k \neq i,j} (a_{ki} - a_{kj})(b_{\pi_k\pi_j} - b_{\pi_k\pi_i}) + (a_{ik} - a_{jk})(b_{\pi_j\pi_k} - b_{\pi_i\pi_k}) \quad (22)$$

Algoritmo 5 Fast descent procedure

1. $I = \emptyset$
2. mientras $|I| < n$ repetir:
   - 2a. elegir i aleatoriamente $1 \leq i \leq n$, $i \notin I$
   - 2b. $J = \{i\}$
   - 2c. mientras $|J| < n$ repetir:
     - 2c1. elegir j, aleatoriamente $1 \leq j \leq n$, $j \notin J$
     - 2c2. si $\Delta(\pi, i, j) < 0$, intercambiar π_i y π_j en π
     - 2c3. $J = J \cup \{j\}$
   - 2d. $I = I \cup \{i\}$

Fast descent procedure examina sistemáticamente todos los posibles intercambios elegidos en orden aleatorio, inmediatamente encuentra uno que mejora la solución inicial lo aplica y repite de nuevo el procedimiento partiendo de la misma solución inicial. Se elige como solución optimizada la solución obtenida en la segunda iteración. En caso de no encontrar un intercambio que mejore la solución, se deja como solución optimizada la misma solución inicial. El procedimiento no necesariamente obtiene un óptimo local por ser first-improvement, pero puede producir diferentes soluciones (dado por el número de iteraciones del procedimiento) partiendo de la misma solución. En HAS-QAP el procedimiento se corre dos veces y por eso sólo puede producir dos soluciones diferentes en el caso que encuentre un intercambio de mejora en la primera iteración.

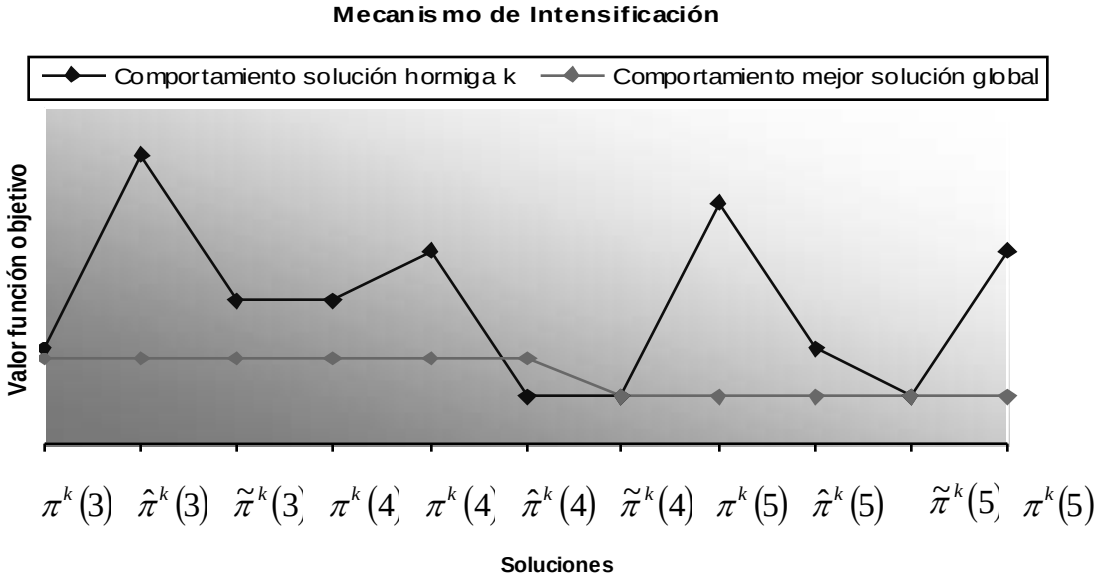


Figura 11. Comportamiento de la solución asociada a una hormiga genérica k de acuerdo al mecanismo de intensificación

Intensificación: Este mecanismo se activa cuando la mejor solución global es mejorada. Cuando esta activado en la iteración t las hormigas artificiales eligen empezar la siguiente iteración $t+1$ con la mejor solución entre $\pi^k(t)$ y $\tilde{\pi}^k(t)$. Por el contrario, cuando no esta activado, las hormigas artificiales simplemente continúan con $\tilde{\pi}^k(t)$. El mecanismo permanece activado si al menos una de las m hormigas artificiales mejora su solución inicial durante la iteración ($f(\tilde{\pi}^k(t)) < f(\pi^k(t))$). La mejor solución global π^k en es actualizada cuando en una determinada iteración, una hormiga k encuentra una solución π^{k*} tal que $f(\pi^{k*}) < f(\pi^*)$ haciendo $\pi^{k*} = \pi^*$. La intensificación enfoca la búsqueda alrededor de la nueva solución π^* mientras que la influencia de su información tenga efecto sobre los rastros de feromona. La figura 12 muestra el comportamiento general de una hormiga artificial k y la variación de sus soluciones de acuerdo al mecanismo de intensificación, empezando por la iteración $t = 3$, frente al comportamiento de la mejor solución global π^*. En la

figura, la hormiga artificial empieza cada iteración con una solución inicial asociada $\pi^k(t)$, cambia después a una solución modificada $\hat{\pi}^k(t)$ y termina con una solución final optimizada con búsqueda local $\tilde{\pi}^k(t)$. La solución inicial de la hormiga artificial en $t=3$ $\pi^k(3)=\pi^*$ y la solución al final de la iteración, la solución optimizada $\tilde{\pi}^k(3)$ aunque mejora a $\hat{\pi}^k(3)$ no supera a la inicial $\pi^k(3)$. Bajo la suposición de que en $t=3$ el mecanismo esta desactivado, la hormiga inicia la siguiente iteración con la solución que obtuvo al final de $t=3$ $\pi^k(4) \leftarrow \tilde{\pi}^k(3)$ y el mecanismo continua desactivado porque no se logra mejorar a π^*. Al final de la iteración $t=4$ la solución $\tilde{\pi}^k(4)$ supera a π^* y la reemplaza $\pi^* \leftarrow \tilde{\pi}^k(4)$; el mecanismo es activado y la hormiga empieza la siguiente iteración con $\tilde{\pi}^k(4)$ ($\pi^k(5) \leftarrow \tilde{\pi}^k(4)$). En $t=5$ la solución $\tilde{\pi}^k(5)$ no es mejor que la inicial, pero como el mecanismo esta activado la hormiga elige a la solución inicial para empezar la próxima iteración $\pi^k(6) \leftarrow \pi^k(5)$.

Actualización de los rastros de feromona: HAS-QAP implementa la actualización fuera de línea utilizando la estrategia elitista mejor-global. Primero se lleva a cabo el procedimiento de *evaporación del feromona* aplicando el coeficiente de decrecimiento del feromona sobre todos los rastros de feromona τ_{ir}. Sólo los rastros de feromona correspondientes a las parejas (i,r) utilizadas en la mejor solución global π^* son reforzados con una cantidad adicional de feromona Δ_{ir}^{gb}. De acuerdo con esto, se identifican dos tipos de actualizaciones en HAS-QAP: la actualización de los rastros de feromona correspondientes a las parejas que no están asociadas a la mejor solución global y en la que sólo esta presente la fase de evaporación del feromona, y la actualización de los rastros de feromona correspondientes a la mejor solución global en la que la fase de evaporación como la de evaporación se implementan paralelamente.

Actualización de los rastros de feromona τ_{ir} *asociadas a las parejas* $(i,r) \notin \pi^*$:

$$\tau_{ir}(t+1) = (1-\rho)\cdot \tau_{ir}(t) \quad \forall (i,r) \notin \pi^* \qquad (23)$$

Actualización de los rastros de feromona τ_{ir} *asociadas a las parejas* $(i,r) \notin \pi^*$:

$$\tau_{ir}(t+1) = (1-\rho)\cdot \tau_{ir}(t) + \Delta_{ir}^{gb} \quad \forall (i,r) \in \pi^* \qquad (24)$$

Diversificación: El mecanismo es activado si durante las últimas S iteraciones el algoritmo no encuentra una mejor solución a π^*. Consiste en borrar toda la información contenida en los rastros de feromona reinicializandolos con un valor τ_0 calculado a partir de $m-1$ soluciones generadas aleatoriamente, más una solución correspondiente a π^* que completa las m soluciones asociadas cada una a una hormiga artificial k.

6.2. COMPONENTES I&D EN HAS-QAP

La intensificación y la diversificación de la búsqueda pueden ser consideradas efectos de los mismos componentes del algoritmo (Blum y Roli [2003]). Todo algoritmo o componente funcional que tenga un efecto de intensificación y/o diversificación en la búsqueda es considerado un componente I&D de acuerdo a la sección 3.2.2. Existen componentes I&D con una sola función y muchos otros con un apropiado balance entre ambas. La tabla 6 reúne los componentes del algoritmo HAS-QAP e identifica para cada uno de ellos los componentes I&D que tienen efecto en la intensificación y o diversificación de la búsqueda.

Componente de HAS-QAP	Componente I&D	Función componente I&D	
		Intensificación	**Diversificación**
Modificación Probabilística	Regla de transición	Función de explotación	Función de exploración
Optimización de soluciones con búsqueda local	Algoritmo fast descent procedure	Evaluación implícita de la función objetivo a través del cálculo de $\Delta(\pi, i, j)$ para determinar si hubo o no mejora de la solución inicial. Esto implica un esfuerzo por encontrar una pareja de posiciones a intercambiar que produzca una solución vecina a la solución inicial con un menor valor de la función objetivo	Elección aleatoria de las parejas de posiciones a intercambiar (elección aleatoria de la solución dentro de un conjunto de soluciones vecinas en la estructura de la vecindad) *Regla pivote first-improvement*:: Elige la primera pareja que produzca una solución dentro de la estructura de la vecindad que mejore la solución inicial. Con ella la búsqueda se realiza en un tiempo computacional reducido que propicia la diversificación.
Intensificación	Asignación de soluciones iniciales a cada hormiga k para la próxima iteración	*Activado* : Asigna a la hormiga k la mejor solución entre la inicial y la final de la iteración actual para iniciar la siguiente iteración	*Desactivado* : Asigna a la hormiga k la solución final de la iteración actual para iniciar la siguiente iteración
Diversificación	Reinicialización de soluciones aleatorias optimizadas con búsqueda local a partir de las cuales se calcula el nuevo valor τ_0 con el que parten los rastros de feromona	—	Reinicialización de la memoria del algoritmo (reinicialización de los valores del rastro de feromona)
Actualización fuera de línea del rastro de feromona	Actualización de la memoria del algoritmo	Reforzamiento de los rastros de feromona	Procedimiento de evaporación del feromona aplicado sobre todos los rastros de feromona

Tabla 6. Identificación de los componentes I&D en cada uno de los componentes del algoritmo HAS-QAP y descripción de la(s) función(es) de intensificación y/o diversificación que cumplen en el proceso de búsqueda

Algoritmo 6. HAS-QAP

Inicialización

Generar m permutaciones iniciales $\pi^1(1),\ldots,\pi^m(1)$ cada una asociada a una hormiga k

Optimizar $\pi^1(1),\ldots,\pi^m(1)$ con fast descent procedure

Hacer π^* la mejor solución global

Inicializar la matriz de rastros de feromona

Activar mecanismo de intensificación

Para cada $t = 1$ hasta $t = I^{\max}$ **repetir**

Manipulación de la solución

Para cada solución $\pi^k(t)$ $(1 \le k \le m)$ hacer

Aplicar R intercambios en la solución $\pi^k(t)$ para obtener $\hat{\pi}^k(t)$

Aplicar fast descent procedure a $\hat{\pi}^k(t)$ para obtener $\tilde{\pi}^k(t)$

Fin para

Intensificación

Para cada hormiga k hacer

Si esta activada, entonces

$\pi^k(t+1) \leftarrow$ mejor solución entre $\pi^k(t)$ y $\tilde{\pi}^k(t)$

de lo contrario $\pi^k(t+1) \leftarrow \tilde{\pi}^k(t)$

Fin para

Si $\forall k \quad \pi^k(t+1) = \pi^k(t)$ entonces

Desactivar intensificación

de lo contrario Sigue Activada

Si $\exists k$ tal que $f(\tilde{\pi}^k(t)) < f(\pi^*)$ entonces

Actualizar la mejor solución global $\pi^* \leftarrow \tilde{\pi}^k(t)$

Activar intensificación

Actualización de los rastros de feromona

$$\tau_{ir}(t+1) = (1-\rho)\cdot\tau_{ir}(t) \quad \forall(i,r) \notin \pi^*$$

$$\tau_{ir}(t+1) = (1-\rho)\cdot\tau_{ir}(t) + \Delta_{ir}^{gb} \quad \forall(i,r) \in \pi^*$$

Diversificación

Si S iteraciones han sido corridas sin mejorar π^* entonces

Activar diversificación

Fin para

Capítulo VII

DISEÑO EXPERIMENTAL

El algoritmo HAS-QAP implementado en este trabajo es el propuesto en [18] corrido con los mismos parámetros empleados por Dorigo y Gambardella. Para diferenciarlo del HAS-QAP original se le ha denominado HAS-QAP^R con el propósito de compararlos frente a un conjunto de instancias de la librería QAPLIB. Los resultados obtenidos con HAS-QAP^R son comparados además con algunos de los mejores algoritmos metaheurísticos disponibles para resolver el problema QAP: los algoritmos de búsqueda tabú, *Búsqueda tabú* (TS) y *Búsqueda tabú reactiva* (RTS), el algoritmo *Genético híbrido-búsqueda tabú* (GH) y el algoritmo de *Enfriamiento simulado* (SA). Las instancias de QAPLIB empleadas para probar el desempeño de HAS-QAP^R han sido clasificadas por Taillard y Dorigo [1997] en dos categorías de acuerdo a la medida estadística *flujo-dominancia*[21] [18] que reúne en un único valor las medidas $fd(B)$ y $dd(A)$. En la tabla 7 se muestran los valores del flujo-dominancia para cada una de las instancias, reportados en [18]. La categoría (1) comprende instancias correspondientes a las clases (c) y (d): problemas reales, irregulares y estructurados, y la categoría (2), a instancias de las clases (a) y (b): problemas generados aleatoriamente, regulares y no estructurados. Taillard [1995] demostró que por lo general para instancias de la clase (c), es decir, problemas de la vida real, la mayoría de los métodos heurísticos y metaheurísticos tienen un pobre desempeño. Por el contrario, los mismos métodos se desempeñan muy bien en problemas clasificados dentro de la categoría (2) con matrices cuyas entradas estén uniformemente distribuidas. Estos métodos aplicados a problemas de este tipo son

[21] (μ_A, μ_B) y (σ_A, σ_B) corresponden a los valores de la media y varianza de los elementos fuera de la diagonal de las matrices A y B respectivamente. Si $\max\left(\sigma_A/\mu_A, \sigma_B/\mu_B\right) > 1{,}2$, el problema es clasificado dentro de la categoría (1), de lo contrario, es clasificado dentro de la (2).

capaces de encontrar soluciones de buena calidad (soluciones hasta con un 1% por debajo de la mejor solución encontrada).

Instancias categoría (1)					
Instancia clase (c)			Instancia clase (d)		
Instancia	n	Flujo-dominancia	Instancia	n	Flujo-dominancia
bur26a	20	2,75	tai20b	20	3,24
bur26b	25	2,75	tai25b	25	3,03
bur26c	30	2,29	tai30b	30	3,18
bur26d	35	2,29	tai35b	42	3,05
bur26d	40	2,55	tai40b	49	3,13
bur26e	50	2,55	tai50b	56	3,10
bur26f	60	2,84	tai60b	64	3,15
bur26g	80	2,84	tai80b	72	3,21
bur26h	50	4,15			
chr25a	25	5,16			
els19	19	1,46			
kra30a	30	1,46			
kra30b	30	3,24			
Instancias categoría (2)					
Instancia clase (a)			**Instancia clase (b)**		
Instancia	n	Flujo-dominancia	Instancia	n	Flujo-dominancia
tai20a	20	0,61	nug20	20	0,99
tai25a	25	0,60	nug30	30	1,09
tai30a	30	0,59	sko42	42	1,06
tai35a	35	0,58	sko49	49	1,07
tai40a	40	0,60	sko56	56	1,09
tai50a	50	0,60	sko64	64	1,07
tai60a	60	0,60	sko72	72	1,06
tai80	80	0,59	sko81	81	1,05
wil50	50	0,64	sko90	90	1,06

Tabla 7. Valores de n y flujo-dominancia para las instancias de QAPLIB elegidas para probar el desempeño de HAS-QAP^R

Dado que todos los algoritmos metaheurísticos contra los que se compara HAS-QAP^R son métodos híbridos que usan algún tipo de algoritmo de búsqueda local, se recalca su influencia en el desempeño final del algoritmo. Específicamente, la influencia del tipo de algoritmo de búsqueda local definido de acuerdo al tipo de instancia del problema QAP a ser resuelto, tal como lo hizo Stützle en [34]. De acuerdo a los

resultados reportados en [18] para las instancias correspondientes a la categoría (2), los métodos híbridos que usan como algoritmo de búsqueda local una búsqueda tabú como el algoritmo GH, y algoritmos de búsqueda tabú como el TS y el RTS demostraron tener el mejor desempeño. Mientras que sus contrapartes los métodos híbridos que emplean como algoritmo de búsqueda local un algoritmo de búsqueda rápida 2-opt como el fast descent procedure: MMAS-QAP$_{2\text{-opt}}$ y HAS-QAP, tienen un menor desempeño que los primeros en estas instancias. En las instancias clasificadas dentro de la categoría (1), especialmente los problemas de la vida real, el desempeño de las características de los algoritmos es diferente. Los algoritmos para resolver este tipo de problemas deben ser capaces de explotar su estructura y guiar la búsqueda local hacia regiones promisorias con soluciones de alta calidad; de acuerdo a la sección 2.2, una vez identificada la estructura es mucho más fácil encontrar tales soluciones. Los algoritmos ACO tienen esta característica aplicados a problemas QAP, pero son precisamente los algoritmos MMAS-QAP$_{2\text{-opt}}$ y HAS-QAP los que tienen el mejor desempeño. La asociación de algoritmos de búsqueda local rápidos como el fast descent procedure, en la influencia sobre los algoritmos metaheurísticos con mejor desempeño en instancias de la categoría (1), motivó el reemplazo del algoritmo de búsqueda tabú en GH por un fast descent procedure que generó el método híbrido GDH propuesto en [25]. Los resultados comprobaron que GDH podía obtener mejores resultados que GH en el mismo tiempo computacional.

7.1. PARÁMETROS DEL ALGORITMO HAS-QAP^R

Los parámetros empleados para implementar y comparar el algoritmo HAS-QAP^R con el resto de métodos son los mismos del algoritmo HAS-QAP de Dorigo y Gambardella [1997], que demostraron ser experimentalmente robustos a pequeñas modificaciones para las instancias definidas anteriormente. La lista de ellos se observa en la tabla 8. Los resultados de un diseño factorial aplicado a los parámetros ρ, S y q en la sección 7.7 indican que los valores dados para ellos en la tabla 8 logran para HAS-QAP^R uno de sus mejores desempeños.

Nombre del parámetro	Parámetro	Valor
Tamaño de la colonia (número de hormigas)	m	10
Número de intercambios aplicados en el mecanismo de modificación	R	$n/3$
Coeficiente de evaporación	ρ	0,1
Coeficiente empleado en el cálculo de τ_0	Q	100
Número de iteraciones necesarias para activar mecanismo de diversificación	S	$n/2$
Coeficiente de determinación de la función a emplear por la regla de transición	q	0,9

Tabla 8. Listado de los parámetros empleados por el algoritmo HAS-QAP^R

7.2. RESULTADOS COMPUTACIONALES

Los resultados computacionales muestran el desempeño de los métodos metaheurísticos propuestos, y el del método HAS-QAP^R en instancias de la categoría 1 y 2 medido para corridas cortas y largas. Para cada instancia todos los algoritmos incluyendo el HAS-QAP^R fueron corridos diez veces (diez réplicas) con un valor máximo de diez y cien iteraciones, corridas cortas y corridas largas respectivamente. Se obtuvo para cada corrida la solución promedio de las diez réplicas. El desempeño corresponde a la fracción de la solución promedio hallada por encima de la mejor solución encontrada.

7.2.1. Comparación de HAS-QAP^R frente a otros algoritmos

Las tablas 9 y 10 dan los resultados computacionales para corridas cortas y largas respectivamente. En corridas cortas se pone a prueba la habilidad de los algoritmos para encontrar buenas soluciones restringidos a un corto tiempo computacional determinado por un número reducido de iteraciones. Para HAS-QAP y HAS-QAP^R las corridas cortas ponen a prueba su desempeño en ausencia de uno de los mecanismos que los caracterizan, el mecanismo de diversificación que sólo se activa después de $n/2$ iteraciones sin mejorar π^*.

	Instancias Categoría (1) Corridas Cortas									
	Instancia	Mejor valor conocido	ID	TS	RTS	SA	GH	HAS-QAP	HAS-QAP-R	cpu(Mils)
Instancias Clase (c)	bur26a	5426670	1	0,2080	.	0,185	0,060	0,027	**0,024**	331
	bur26b	3817852	2	0,4410	.	0,191	0,090	0,106	**0,025**	71
	bur26c	5426795	3	0,1700	.	0,137	**0,004**	0,009	0,025	40
	bur26d	3821225	4	0,2490	.	0,379	0,003	**0,002**	0,035	331
	bur26e	5386879	5	0,0760	.	0,228	**0,003**	0,004	0,028	301
	bur26f	3782044	6	0,3690	.	0,224	0,006	**0,000**	0,030	361
	bur26g	10117172	7	0,0780	.	0,139	0,006	**0,000**	0,025	320
	bur26h	7098658	8	0,3490	.	0,368	0,003	**0,001**	0,028	260
	chr25	3796	9	15,9690	16,844	27,139	15,158	15,690	**2,035**	271
	els19	17212548	10	21,2610	6,714	16,028	0,515	0,923	**0,488**	241
	Kr30a	88900	11	2,6660	2,155	1,813	1,576	1,664	**0,284**	330
	Kr30b	91420	12	0,4780	1,061	1,065	0,451	0,504	**0,279**	270
Instancias Clase (d)	Tai20b	122455319	13	6,7000	.	14,392	**0,150**	0,243	0,177	251
	Tai25b	344355646	14	11,4860	.	8,831	0,874	**0,133**	0,413	260
	Tai30b	637117113	15	13,2840	.	13,515	0,952	**0,260**	0,385	320
	Tai35b	283315445	16	10,1650	.	6,935	1,084	**0,343**	0,370	331
	Tai40b	637250948	17	9,6120	.	5,430	1,621	**0,280**	0,434	361
	Tai50b	458821517	18	7,6020	.	4,351	1,397	**0,291**	0,423	541
	Tai60b	608245054	19	8,6920	.	3,678	2,005	**0,313**	0,409	621
	Tai80b	818415043	20	6,0080	.	2,793	2,643	1,108	**0,377**	791

Tabla 9a. Desempeño del algoritmo HAS-QAP^R en corridas cortas para Instancias de la categoría 1

	Instancias Categoría (2) Corridas cortas									
	Instancia	Mejor valor conocido	ID	TS	RTS	SA	GH	HAS-QAP	HAS-QAP-R	cpu(Mils)
Instancias Clase (a)	tai20a	703482	1	0,769	0,705	1,209	0,732	1,483	**0,1382**	261
	tai25a	1167256	2	1,128	0,892	1,766	1,371	2,527	**0,1408**	251
	tai30a	1818146	3	0,871	1,044	1,434	1,160	2,600	**0,1348**	261
	tai35a	2422002	4	1,356	1,192	1,886	1,455	2,969	**0,1460**	301
	tai40a	3139370	5	1,284	0,996	1,750	1,590	3,063	**0,1468**	320
	tai50a	4941410	6	1,377	1,241	2,296	1,841	3,487	**0,1496**	451
	tai60a	7208572	7	1,544	1,248	1,942	1,867	3,686	**0,1445**	521
	tai80a	13557864	8	1,170	0,749	1,773	1,344	2,996	**0,1272**	832
	Wil50	48816	9	**0,041**	0,504	0,149	0,253	0,211	0,0946	410
Instancias Clase (b)	Nug 20	2570	10	**0,101**	0,911	0,327	0,047	0,156	0,1393	261
	Nug 30	6124	11	0,271	0,872	0,500	0,249	0,565	**0,1877**	280
	Sko 42	15812	12	0,187	1,116	0,301	0,477	0,654	**0,1806**	460
	Sko 49	23386	13	0,198	0,978	0,406	0,368	0,661	**0,1678**	510
	Sko56	34458	14	0,347	1,082	0,504	0,515	0,729	**0,1709**	581
	Sko64	48498	15	0,221	0,861	0,390	0,631	0,504	**0,1615**	580
	Sko72	66256	16	0,478	0,948	0,323	0,616	0,702	**0,1546**	671
	Sko81	90998	17	0,304	0,880	0,289	0,628	0,493	**0,1481**	771
	Sko90	115534	18	0,386	0,748	0,418	0,632	0,591	**0,1466**	962

Tabla 9b. Desempeño del algoritmo HAS-QAP^R en corridas cortas para Instancias de la categoría 2

La última columna corresponde al tiempo computacional empleado por HAS-QAP^R dado en milisegundos. El mejor desempeño (fracción) para cada instancia está resaltado en negrilla. En general, a simple vista, el desempeño de los métodos muestra una dependencia significativa del tipo de instancia, tanto dentro de los grupos (instancias de la misma categoría) como entre los grupos (instancias de diferentes categorías). Los valores de la tabla 9a demuestran la habilidad de HAS-QAP^R en instancias de la categoría 1 para obtener buenas soluciones en corridas cortas superando la de HAS-QAP en las instancias de la clase (c), bur26a, bur26b, chr25, els19, kr30a y kr30b (instancias de la vida real), y en la instancia tai80b de la clase (d), en el resto de instancias su desempeño esta casi al mismo nivel del de HAS-QAP y GH. GH sólo supera en corridas cortas a HAS-QAP en las instancias bur26e y tai20b. Los desempeños obtenidos por HAS-QAP en corridas largas (tabla 10a) indican que es el mejor dentro de esta categoría. En las instancias bur26 obtiene el mejor valor conocido al igual que GH, excepto por la bur26a y la bur26b; y en las instancias taixb lo obtiene para $n = 25$, 30 y 40. GH supera en corridas largas a HAS-QAP^R en el resto de instancias de la clase (c) menos en la chr25. HAS-QAP^R es el mejor método en las instancias de la categoría 2 con desempeños muy superiores al del resto, en corridas cortas, en corridas largas, sin embargo, los desempeños que obtiene para la clase b están al mismo nivel de los otros. Le sigue el método TS, y el RTS sólo en instancias de la clase (a). De último están los métodos SA y GH. En corridas largas GH se pone al nivel de TS superándolo en algunas instancias, lo que demuestra su competencia junto con HAS-QAP^R resolviendo instancias de ambas categorías. El método SA es el peor dentro de esta categoría con los peores desempeños en instancias de la clase (a), pero que superan a los de RTS en la clase (b).

	Instancias Categoría (1) Corridas largas									
	Instancia	Mejor valor conocido	ID	TS	RTS	SA	GH	HAS-QAP	HAS-QAP-R	cpu(Mils)
Instancias Clase (c)	Bur26a	5426670	1	0,0004	.	0,141	0,012	**0,000**	0,0070	1570
	Bur26b	3817852	2	0,0032	.	0,183	0,022	**0,000**	0,0069	1552
	Bur26c	5426795	3	0,0004	.	0,074	**0,000**	**0,000**	0,0067	1573
	Bur26d	3821225	4	0,0015	.	0,006	**0,000**	**0,000**	0,0070	1572
	Bur26e	5386879	5	0,0000	.	0,124	**0,000**	**0,000**	0,0007	1573
	Bur26f	3782044	6	0,0007	.	0,158	**0,000**	**0,000**	0,0004	1442
	Bur26g	10117172	7	0,0003	.	0,169	**0,000**	**0,000**	0,0007	1522
	Bur26h	7098658	8	0,0027	.	0,127	**0,000**	**0,000**	0,0005	1482
	Chr25	3796	9	6,9652	9,8894	12,497	2,692	3,082	**1,2528**	1432
	Els19	17212548	10	0,0000	0,0899	18,539	**0,000**	**0,000**	0,0024	1062
	kr30a	88900	11	0,4702	2,0079	1,466	**0,134**	0,630	0,1627	1823
	kr30b	91420	12	0,0591	0,7121	0,195	**0,054**	0,071	0,1546	1692
Instancias Clase (d)	tai20b	122455319	13	**0,0000**	.	6,730	**0,000**	0,091	0,0349	1202
	tai25b	344355646	14	0,0072	.	1,122	**0,000**	**0,000**	0,1153	1392
	tai30b	637117113	15	0,0547	.	4,408	**0,000**	**0,000**	0,1362	1813
	tai35b	283315445	16	0,1777	.	3,175	0,107	**0,026**	0,0271	2103
	tai40b	637250948	17	0,2082	.	4,565	0,211	**0,000**	0,1342	2573
	tai50b	458821517	18	0,2943	.	0,811	0,214	**0,192**	0,2534	3314
	tai60b	608245054	19	0,3904	.	2,137	0,291	**0,048**	0,2682	4326
	tai80b	818415043	20	1,4354	.	1,439	0,829	0,667	**0,2956**	6709

Tabla 10a. Desempeño del algoritmo HAS-QAP^R en corridas largas para Instancias de la categoría 1

	Instancias Categoría (2) Corridas largas									
	Instancia	Mejor valor conocido	ID	TS	RTS	SA	GH	HAS-QAP	HAS-QAP-R	cpu(Mils)
Instancias Clase (a)	Tai20a	703482	1	0,211	0,246	0,716	0,268	0,675	**0,0860**	1152
	Tai25a	1167256	2	0,510	0,345	0,716	0,629	1,189	**0,1002**	1382
	Tai30a	1818146	3	0,340	0,286	1,002	0,439	1,311	**0,0913**	1743
	Tai35a	2422002	4	0,757	0,355	0,907	0,698	1,762	**0,1159**	2093
	Tai40a	3139370	5	1,006	0,623	1,345	0,884	1,989	**0,1229**	2414
	Tai50a	4941410	6	1,145	0,834	1,539	1,049	2,800	**0,1270**	3305
	Tai60a	7208572	7	1,270	0,831	1,395	1,159	3,070	**0,1090**	4176
	Tai80a	13557864	8	0,854	0,467	0,995	0,796	2,689	**0,1135**	6419
	wil50	48816	9	0,041	0,504	0,061	**0,032**	0,061	0,0704	3264
Instancias Clase (b)	Nug 20	2570	10	**0,000**	0,911	0,070	0,000	**0,000**	0,0602	1142
	Nug 30	6124	11	0,032	0,872	0,121	**0,007**	0,098	0,0539	1802
	sko 42	15812	12	0,039	1,116	0,114	**0,003**	0,076	0,1073	2714
	sko 49	23386	13	0,062	0,978	0,133	**0,040**	0,141	0,0948	3294
	sko56	34458	14	0,080	1,082	0,110	**0,060**	0,101	0,1399	3876
	sko64	48498	15	**0,064**	0,861	0,095	0,092	0,129	0,1315	4767
	sko72	66256	16	0,148	0,948	0,178	0,143	0,277	**0,1083**	5788
	sko81	90998	17	**0,098**	0,880	0,206	0,136	0,144	0,1167	6820
	sko90	115534	18	0,169	0,748	0,227	0,196	0,231	**0,1295**	7571

Tabla 10b. Desempeño del algoritmo HAS-QAP^R en corridas largas para Instancias de la categoría 2

Los valores de la última columna indican que el tiempo computacional que gasta HAS-QAP^R para resolver una instancia esta en función de su tamaño. Por ejemplo, en las instancias bur26 tarda aproximadamente el mismo tiempo entre ellas, en la els19 emplea menos tiempo y en las kr30 los tiempos son parecidos a las primeras. En las taixxa, taixxb, y las skox se observa el incremento proporcional del tiempo con el aumento del tamaño xx. Los tiempos computacionales dados en [18] para el algoritmo HAS-QAP tienen en general el mismo comportamiento de los de HAS-QAP^R para todas las instancias.

Capítulo VIII

APLICACIÓN DE TÉCNICAS ESTADÍSTICAS MULTIVARIADAS PARA LA IDENTIFICACIÓN DE INTERRELACIONES ENTRE LOS MÉTODOS

La aplicación de las técnicas estadísticas multivariadas anova, componentes principales y factores tuvo como propósito investigar las interrelaciones entre los métodos basándose en sus desempeños, con el fin de agruparlos.

Los resultados obtenidos con las técnicas componentes principales y factores fueron congruentes, como era esperado. Establecieron los mismos niveles de asociación entre los métodos. La agrupación de los métodos obedeció al mismo conjunto de variables (métodos) asociados a cada componente principal, y a cada factor común, como se verá más adelante.

8.1. PRUEBA DE IGUALDAD DE MEDIAS

El análisis anova evaluó la influencia del tipo de instancia a ser resuelta sobre el desempeño de los métodos, las diferencias entre los promedios de los métodos, y las diferencias entre los promedios de las instancias. Luego se compararon las medias de todos los tratamientos con el fin de identificar los mejores métodos para cada una de las categorías. El procedimiento empleado fue el *Proc anova* del programa SAS.

8.1.1. Prueba de igualdad de medias en la categoría 1

Las tablas que se observan en los Anexos C y D corresponden a los resultados estadísticos arrojados por el procedimiento proc anova corrido para las instancias de la categoría 1. La variable tratamientos corresponde a los cinco métodos probados, y

la variable bloque a las instancias. La primera tabla muestra el efecto conjunto bloque-tratamiento, la segunda el efecto tratamiento y por último el efecto bloque. El efecto bloque-tratamiento con un p-value de <.0001, es significativo, demostrando que el desempeño alcanzado por los métodos en corridas cortas varía en función de la instancia a ser resuelta. El efecto independiente de bloques y tratamientos es significativo con un p-value de <.0001 y 0.0004 respectivamente, por tanto, existen diferencias significativas entre los promedios de los métodos, y los promedios de las instancias. Esto significa que hay mejores tratamientos que otros dentro de una misma categoría, y que hay instancias más difíciles de resolver que otras. En corridas largas los tres efectos resultaron ser igual de significativos con valores de p-value de 0.0003, 0.0577 y 0.0007 respectivamente. En la tabla 11a se observan los p-values asociados a cada efecto para ambas corridas.

	p-value		
Resultados Corrida	Efecto trat-bloque	Efecto trat	Efecto bloque
Cortas	<.0001	<.0001	0.0004
Largas	0.0003	0.0577	0.0007

Tabla 11a. P-values de los efectos asociados a instancias de la categoría 1

Tratamiento	Método	Media Corridas Cortas	Media Corridas Largas
1	TS	5,79315	0,50358
2	RTS	6,69350	3,17483
3	SA	5,39105	2,90309
4	GH	1,43005	0,22827
5	HAS-QAP	1,09505	0,24031
6	HAS-QAP-R	0,31470	0,14337

Tabla 11b. Media de los tratamientos en instancias de la categoría 1

8.1.2. Prueba de igualdad de medias en la categoría 2

Los Anexos E y F corresponden a los resultados estadísticos arrojados por proc anova para las instancias de la categoría 2. En corridas cortas el efecto bloque-tratamiento sobre el desempeño es significativo con un p-value de <.0001. Por otro lado, con p-values de <.0001 y <.0001 respectivamente, el efecto independiente de bloques y tratamientos también es significativo. Los efectos en corridas largas son igualmente

significativos con p-values de <.0001, 0.0005 y <.0001. Los p-values asociados a éste tipo de instancias se resumen en la tabla 12a.

	p-value		
Resultados Corrida	Efecto trat-bloque	Efecto trat	Efecto bloque
Cortas	<.0001	<.0001	<.0001
Largas	<.0001	0,0005	<.0001

Tabla 12a. P-values de los efectos asociados a instancias categoría 2

Tratamiento	Método	Media Corridas Cortas	Media Corridas Largas
1	TS	0,66850	0,37922
2	RTS	0,94261	0,71594
3	SA	0,98128	0,55167
4	GH	0,87644	0,36839
5	HAS-QAP	1,55983	0,93017
6	HAS-QAP-R	0,14886	0,10435

Tabla 12b. Media de los tratamientos en instancias de la categoría 2

El procedimiento Proc anova arrojó las medias correspondientes a cada tratamiento. El análisis anova es equivalente a una prueba de igualdad de medias, por tal razón, si el efecto independiente de los tratamientos es significativo, las medias entre métodos deben diferir significativamente. La comparación de las medias entre corridas confirma lo que a simple vista se observa en las tablas 9a, 9b, 10a y 10b, que los desempeños mejoran en corridas largas. Las tablas 11b y 12b contienen el listado de las medias de cada tratamiento para ambas corridas correspondientes a la categoría 1 y categoría 2 respectivamente. Los valores que se observan en la tabla 11b demuestran la superioridad del método HAS-QAP^R en instancias de la categoría 1 para corridas largas y cortas, con los valores mínimos: 0,31470 y 0,14337 respectivamente. Le siguen los algoritmos HAS-QAP y GH con medias muy parecidas, que los identifican como algoritmos afines. De último se encuentra el algoritmo SA, seguido por TS, con medias también muy parecidas en corridas cortas, diferente a lo que sucede en corridas largas donde TS supera por mucho a SA que tiene asociado el valor máximo. Por insuficiencia de información (ausencia de datos para la mayoría de las instancias), el método RTS no fue tomado en cuenta en el análisis anova y su media corresponde al promedio de los desempeños obtenidos para

las instancias chr25, els19 y las kr30. En instancias de la categoría 2, HAS-QAP^R es muy superior al resto de métodos con un promedio significativamente bajo. Después de él, los algoritmos TS y GH tienen los promedios más bajos. Seguidamente están los algoritmos RTS y SA con promedios similares. El peor método dentro de esta categoría resultó ser HAS-QAP con un promedio significativamente alto. De esta forma, queda confirmada la competencia del algoritmo GH para resolver ambos tipos de instancias, y se demuestra la del algoritmo implementado HAS-QAP^R, con el mejor desempeño en promedio dentro de la categoría 1, y en la categoría 2 uno que supera al de los métodos de búsqueda tabú, TS y RTS, y al de GH, considerados los mejores hasta el momento dentro de ella.

8.2. ANÁLISIS DE COMPONENTES PRINCIPALES

El análisis de componentes principales agrupó las instancias de acuerdo al desempeño alcanzado por los métodos resolviéndolas. En este trabajo, el objetivo de este análisis fue reducir el número de variables (métodos) que explican los resultados (desempeños) a un número reducido de ellas conocidas como componentes principales que lo hicieran sin una pérdida sustancial de información. De esta forma, se buscó explicar la variabilidad de los desempeños obtenidos para las instancias, por sólo unos cuantos componentes principales (un número menor al total de los métodos) asociados cada uno a varios de los métodos. El número de componentes principales retenidos está determinado por el valor de sus eigenvalores. Se retuvieron únicamente los componentes que son estadísticamente significativos, es decir, aquellos con valores del eigenvalor mayores o iguales a uno.

8.2.1. Análisis de componentes principales en la categoría 1

El número de componentes principales retenidos tanto en corridas cortas como largas en las instancias de la categoría 1 fueron dos: Prin1 y Prin2, de acuerdo al valor de los eigenvalores dados en la tabla 13a. Esto significa que los desempeños de los cinco métodos pueden ser explicados en un 0,9737% en corridas cortas y en un 0,9857% en

corridas largas (valores sombreados) por sólo dos componentes principales (variables Prin). De los porcentajes anteriores, la variable Prin1 aporta el 0,8086% y el 0,8324% respectivamente, el resto, lo que dejó de ser explicado por Prin1, lo explica la variable Prin2 con un 0,1651% y un 0,1534%.

Corridas	Variable Prin	Valor del Eigenvalor	Proporción de la varianza explicada	Acumulado
Cortas	**1**	4,04297	0,8086	0,8086
	2	0,82565	0,1651	0,9737
Largas	**1**	4,16177	0,8324	0,8324
	2	0,76691	0,1534	0,9857

Tabla 13a. Variables Prin retenidas y porcentajes de la varianza explicada en las instancias de la categoría 1

	Corridas Cortas		Corridas Largas	
Tratamiento	Prin1	Prin2	Prin1	Prin2
1	0,364753	**0,728074**	0,486134	-0,11323
3	0,451495	0,361917	0,262312	**0,964627**
4	0,464022	-0,382181	0,483858	-0,134607
5	0,456344	-0,422355	0,481852	-0,127916
6	**0,48943**	-0,120325	**0,478085**	-0,148971

Tabla 13b. Valores de los variables Prin asociadas a cada tratamiento en las instancias de la categoría 1

Los pesos (valores de las variables Prin) que tienen los métodos (tratamientos) en la formación de cada variable Prin se observan en la tabla 13b. En corridas cortas la formación de Prin1 no está dominada por ninguno, y por tanto, corresponde a una suma ponderada de los pesos de todos los métodos. Por el contrario, en Prin2 el método con mayor participación en su conformación es TS, correspondiente al tratamiento uno, y con un peso promedio, el método SA correspondiente al tratamiento tres. En corridas largas sucede algo parecido, la formación de la variable Prin1 no esta dominada por ningún tratamiento, mientras que la variable Prin2 lo esta por el método SA. El menor peso en Prin1 lo tiene HAS-QAP^R, el resto de métodos, TS, GH y HAS-QAP tienen pesos muy parecidos. El mayor peso en Prin2 lo tiene el método SA. De esa forma, la variable Prin1 en corridas cortas y largas, identifica las instancias que pueden ser resueltas por todos los métodos, con desempeños similares.

La variable Prin2 en corridas cortas identifica aquellas instancias en las que los métodos TS y SA tienen los peores desempeños, y en corridas largas, aquellas en las que el método SA tiene los peores. La representación gráfica de las instancias dentro de cada variable Prin se observa en la figura 12.

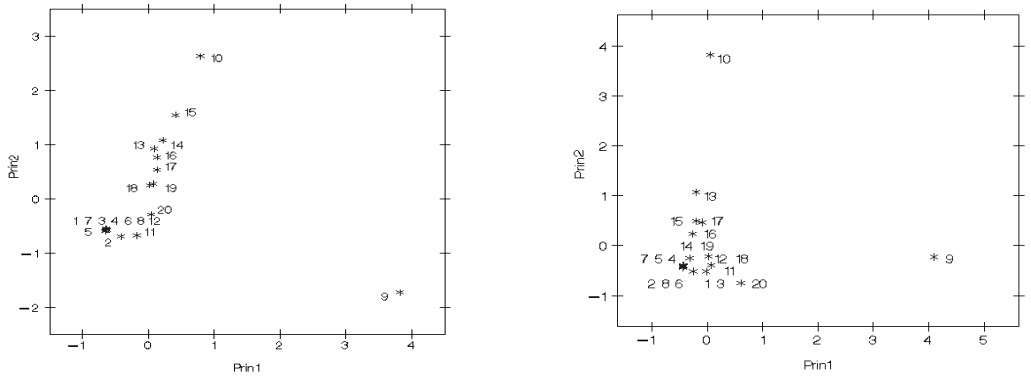


Figura 12. Agrupación de las instancias de la categoría 1 de acuerdo a las variables Prin en corridas cortas (izquierda) y largas (derecha)

La clasificación de las instancias de la categoría 1 dentro de los grupos definidos por las variables Prin en corridas cortas y largas se observa en la tabla 14.

	Corridas Cortas	**Corridas Largas**
Grupo	Instancias	Instancias
1	10,15,14,16,17,19,18,13,20	10,13,17,15,16,19,14
2	1,7,3,4,6,8,5,2	7,5,4,2,8,6
3	11,12	11,12,18,20
4	9	9

Tabla 14. Clasificación de las instancias de la categoría 1 dentro de los grupos definidos por las variables Prin en corridas cortas y largas

La definición de los grupos se da a continuación.

Grupo 1: Instancias donde los métodos TS y SA tienen los peores desempeños

Grupo 2: Instancias donde todos los métodos tienen buenos desempeños y muy parecidos entre ellos

Grupo 3: Instancias con bajos desempeños alcanzados por todos los métodos, siendo los de TS y SA, los peores.
Grupo 4: Instancias con los peores desempeños alcanzados por todos los métodos (las más difíciles de resolver)

En corridas largas el grupo 1 sólo esta definido por el método SA, que es también el único que cuenta en el grupo 3, ya no el TS. En ellas, casi todos los métodos incluyendo el TS obtienen buenos desempeños, de hecho este método los mejora significativamente con relación a los que obtiene en corridas cortas, a diferencia del método SA que queda confirmado como el peor método para resolver las instancias de esta categoría. De acuerdo con esta clasificación, las instancias bur26 son las más fáciles de resolver, con soluciones que no están más de un 0.7% por debajo de la mejor encontrada. En las instancias kr30a y kr30b, y las tai50b y tai80b, el desempeño de los métodos es inferior al alcanzado para las bur26, y los métodos TS y SA tienen los peores. Para el resto de instancias (**Grupo 3**) las soluciones obtenidas por los métodos TS y SA son hasta 16 veces la mejor solución encontrada, demostrando su incapacidad para resolver instancias de esta categoría. La instancia más difícil de resolver es la 9 (chr25), con los desempeños más bajos alcanzados por todos los métodos, las soluciones obtenidas son hasta 27 veces la mejor encontrada.

8.2.2. Análisis de componentes principales en instancias de la categoría 2

El número de variables Prin retenidas tanto en corridas cortas como largas en las instancias de la categoría 2 fueron dos: Prin1 y Prin2. La razón es que con sólo dos componentes principales los desempeños pueden ser representados sin una pérdida sustancial de información (Acumulados superiores al 80%). Con base a la tabla 16a, los desempeños obtenidos por los cinco métodos pueden ser explicados en un 0,947% en corridas cortas y en un 0,8971% en corridas largas (valores sombreados) por las dos variables Prin. De los porcentajes anteriores, la variable Prin1 aporta el 0,7071%

y el 0,6985% respectivamente, el resto, lo que dejó de ser explicado por Prin1, lo explica la variable Prin2 con un 0,2399% y 0,1985%.

Corridas	Variable Prin	Valor del Eigenvalor	Proporción de la varianza explicada	Acumulado
Cortas	**1**	4,24252	0,7071	0,7071
	2	1,43923	0,2399	0,947
Largas	**1**	4,1912638	0,6985	0,6985
	2	1,1910438	0,1985	0,8971

Tabla 15a. Variables Prin retenidas y porcentajes de la varianza explicada en las instancias de la categoría 2

	Corridas Cortas		Corridas Largas	
Trat	Prin1	Prin2	Prin1	Prin2
1	0,47901	-0,058411	0,478581	0,079316
2	**0,283525**	**0,609521**	**-0,216484**	**0,6784**
3	0,474214	-0,078125	0,476996	-0,071146
4	0,476223	-0,035414	0,484655	0,026312
5	0,481434	-0,055401	0,478823	-0,001485
6	**-0,081953**	**0,783993**	**0,18002**	**0,726504**

Tabla 15b. Valores de los variables Prin asociadas a cada tratamiento en las instancias de la categoría 2

Los pesos que recibieron los métodos están dados en la tabla 15b. En corridas cortas y largas la variable Prin1 corresponde a la suma ponderada de los pesos de todos los métodos, excepto el HAS-QAP^R con un peso significativamente bajo y negativo. En la variable Prin2 los métodos con mayor participación en su conformación son el RTS y el HAS-QAP^R. De esta forma, la variable Prin1 en corridas cortas y largas permiten identificar las instancias en la medida en que pueden ser resueltas por los métodos TS, SA, GH y HAS-QAP, con desempeños similares. La variable Prin2 las agrupa de acuerdo a la habilidad que tengan los métodos HAS-QAP^R y RTS para resolverlas. La representación gráfica de las instancias dentro de cada variable Prin se observa en la figura 13.

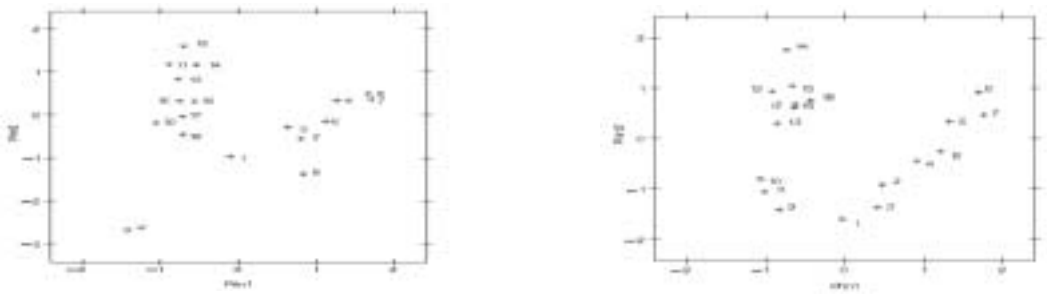


Figura 13. Agrupación de las instancias de la categoría 2 dentro de cada variable Prin en corridas cortas (izquierda) y largas (derecha)

Las tablas 16 y 17 dan la clasificación de las instancias de la categoría 2 de acuerdo a las variables Prin en corridas cortas y largas respectivamente.

	Instancias	
Clasificación	(Clase a)	(Clase b)
A : Bajos desempeños obtenidos por los métodos TS, SA, GH y HAS-QAP y bajos desempeños obtenidos por HAS-QAP^R y RTS	4,6, 7,	10, 15, 16, 17, 18
B : Altos desempeños obtenidos por los métodos TS, SA, GH y HAS-QAP y altos desempeños obtenidos por HAS-QAP^R y RTS	1,2, 3,5,8	11, 12, 13, 14
C : Instancias con los mejores desempeños obtenidos por todos los métodos, incluyendo el HAS-QAP^R y el RTS	9	-

Tabla 16. Clasificación de las instancias de la categoría 2 dentro de los grupos definidos por las variables Prin en corridas cortas

	Instancias	
Clasificación	(Clase a)	(Clase b)
A : Bajos desempeños obtenidos por los métodos TS, SA, GH y HAS-QAP y bajos desempeños obtenidos por HAS-QAP^R y RTS	5,6, 7	18, 17, 16
B : Altos desempeños obtenidos por los métodos TS, SA, GH y HAS-QAP y altos desempeños obtenidos por HAS-QAP^R y RTS	1,3, 2,4,8	12, 13, 14, 15
C : Instancias con los mejores desempeños obtenidos por todos los métodos, incluyendo el HAS-QAP^R y el RTS	9	10, 11

Tabla 17. Clasificación de las instancias de la categoría 2 dentro de los grupos definidos por las variables Prin en corridas largas

En corridas cortas los desempeños obtenidos para las instancias de la clase (a) con los métodos TS, SA, GH y HAS-QAP son soluciones hasta 4 veces la mejor solución encontrada. En ellas, el método HAS-QAP^R tiene los mejores desempeños, con soluciones que no pasan del 13% por debajo de la mejor encontrada. La instancia 9 (wil50) es la instancia más fácil de resolver, con soluciones que no van más allá del 50%, y donde la mejor, sólo está un 4% por debajo. En las instancias de la clase (b) la gran mayoría de las soluciones obtenidas con los métodos TS, SA, GH y HAS-QAP no tienen más de un 70% por debajo de la mejor encontrada, y por tanto son las más fáciles de resolver dentro de esta categoría. En ellas, HAS-QAP^R tiene los peores desempeños, y aún así sus soluciones no pasan del 19% por debajo de la mejor encontrada.

En corridas largas se mantiene casi la misma clasificación dada para corridas cortas. De acuerdo con ella, las instancias más fáciles de resolver son la 9 (wil50) y las nug (20 y 30), la primera dentro de la clase a y las segundas dentro de la b. El método HAS-QAP^R queda confirmado como el mejor método dentro de esta categoría, con soluciones que no están más del 13% por debajo de la mejor encontrada.

8.3. ANÁLISIS DE FACTORES

El propósito de este análisis es agrupar los métodos de acuerdo a los desempeños que obtienen para las instancias de ambas categorías, basándose en sus intercorrelaciones (valores de la matriz de correlación). La matriz de correlaciones es transformada a un modelo de estimación con variables denominadas factores. El número de factores a retener esta determinado, al igual que en componentes principales, por aquellos factores con eigenvalores mayores o iguales a uno. Valores pequeños de la matriz de correlación de residuales y medidas de kaiser mayores a 0,7 demostraron que la

estructura (variabilidad) de los desempeños podía estar bien explicada por el modelo de factores. Los Anexos F, G, H e I contienen el listado completo de los resultados del análisis de factores en ambas categorías.

8.3.1. Análisis de factores en la categoría 1

Los resultados que se observan en la tabla 18a corresponden al análisis de factores no rotados y con rotación varimax en instancias de la categoría 1 para corridas cortas. De acuerdo a la regla de los eigenvalores el número de factores extraídos para cada análisis fueron dos, Factor1 y Factor2. La columnas tres, cuatro, cinco y seis contienen las cargas (lo que en componentes principales se conoce como peso) que determinan cuan bien esta explicada la variable (método) por cada uno de los factores. Los valores de las tres últimas filas corresponden a los eigenvalores de los factores (suma de las cargas), que indican la importancia relativa de cada factor en la explicación de la varianza de los desempeños, a los porcentajes de la varianza explicada por cada factor, y al total de la varianza explicada. Con un total de la varianza explicada del 99,78% en corridas cortas y un 99,79% en corridas cortas, se puede decir que los desempeños están significativamente representados por sólo dos factores, Factor 1 y Factor 2.

		Análisis de Factores No Rotados		Análisis de Factores con rotación varimax	
Trat	Métodos	Factor 1	Factor 2	Factor 1	Factor 2
6	HAS-QAP-R	0,98601	-0,0907	0,84678	0,51323
4	GH	0,93646	-0,33578	0,95263	0,2867
5	HAS-QAP	0,91902	-0,3694	0,98601	0,2493
3	SA	0,88935	0,32047	0,52462	0,78639
1	TS	0,72004	0,63657	0,20057	0,93992
Eigenvalores		4,0031593	0,76533566	2,8589043	1,9096117
Porcentaje varianza		83,77	16,01	59,74	40,04
Total porcentaje varianza		99,78			

Tabla 18a. Resultados del análisis de factores no rotados y con rotación varimax en instancias de la categoría 1 para corridas cortas

La interpretación de los factores está basada en sus cargas incluyendo sus signos. En el análisis de factores no rotados para corridas cortas, las cargas del Factor 1 son significativamente altas, entre mayor valor de la carga, mayor participación del método dentro del factor. En el Factor 2 sólo dos métodos tienen cargas significativas, que son mayores en el Factor 1, el resto, tiene asociado cargas pequeñas y negativas. Con base a lo anterior, la clasificación de los métodos en corridas cortas puede estar dada únicamente por el Factor1. En él identificamos dos tipos de métodos. En el primero están los métodos GH, HAS-QAP y HAS-QAP^R (métodos del tipo 1). En el segundo están los métodos TS (tratamiento 1) y SA (tratamiento 3) (métodos del tipo 2). El método RTS (tratamiento 2) no fue tenido en cuenta en este análisis por las mismas razones que no se le tuvo en los anteriores. El análisis de factores con rotación varimax hizo más evidente la distinción de los dos tipos de métodos, con una mejor distribución de las cargas dentro de cada factor. En él, el Factor1 esta directamente asociado a los métodos GH, HAS-QAP y HAS-QAP^R y el Factor 2 a los métodos TS y SA. La figura 14a muestra la ubicación de los métodos dentro de los factores no rotados, y con rotación varimax.

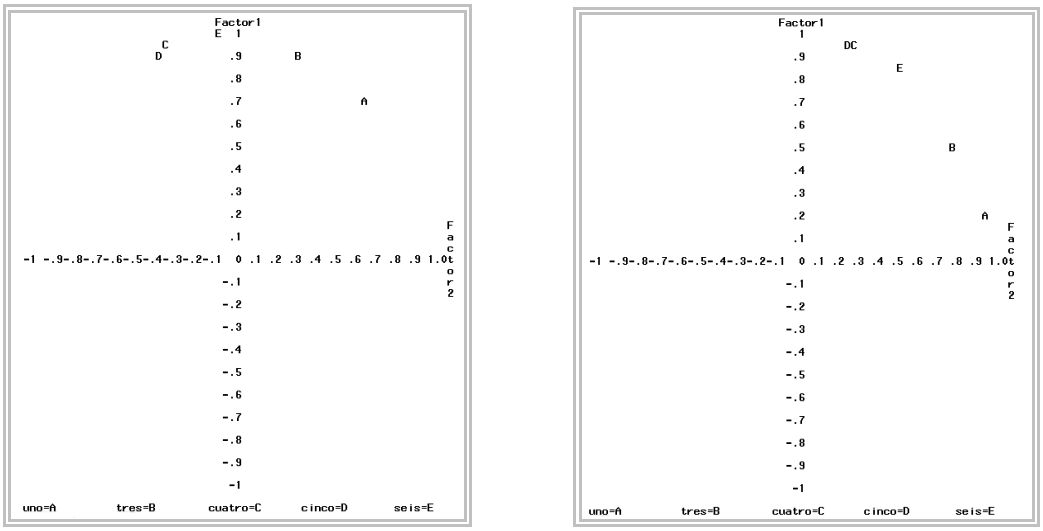


Figura 14a. Representación gráfica de los métodos dentro de los factores no rotados (izquierda) y con rotación varimax (derecha) en instancias de la categoría 1 para corridas cortas

La afinidad de los métodos GH y HAS-QAP se observa claramente en la figura 14 por su cercanía, y en la tabla 18a, por los valores muy similares de sus cargas en el factor 1 que reflejan su alta correlación. El método HAS-QAP^R aunque pertenece al tipo 1 junto con los anteriores, su correlación con ellos es menor, dada por un valor de la carga inferior al de GH y HAS-QAP. Lo anterior indica que tiene una estructura (variabilidad) de sus desempeños diferente a la del algoritmo original HAS-QAP, y por ende un comportamiento diferente frente a la solución del mismo tipo de instancias. En la figura esto se refleja porque se ubica en un punto más alejado de los primeros (GH y HAS-QAP). Los métodos TS y SA conforman el segundo tipo de métodos, con desempeños inferiores a los del tipo 1 que los ubica por debajo de ellos en la figura de los factores con rotación varimax.

En corridas largas la interpretación de las cargas de los factores midió más la habilidad de los métodos para obtener buenos desempeños, que la medida en que se distinguen entre ellos como miembros de diferentes tipos. Las corridas largas ponen a prueba a los algoritmos en cuanto puedan lograr mejoras significativas de sus desempeños frente a los que obtienen en corridas cortas. De ese modo, el análisis de factores en corridas largas determinó la permanencia de los algoritmos GH, HAS-QAP y HAS-QAP^R como los mejores en esta categoría, y el acercamiento de los algoritmos del tipo 2 al nivel de los tipo 1, medido por la mejora en sus desempeños, con respecto a los alcanzados en corridas cortas.

		Análisis de Factores No Rotados		Análisis de Factores con rotación varimax	
Trat	Métodos	Factor 1	Factor 2	Factor 1	Factor 2
1	TS	0,99903	0,03451	0,80177	0,59159
4	GH	0,99347	-0,04525	0,84259	0,52827
5	HAS-QAP	0,98707	-0,00029	0,81175	0,56159
6	HAS-QAP-R	0,97206	-0,07368	0,84117	0,49270
3	SA	0,43894	0,18770	0,25407	0,40417
Eigenvalores		4,096030	0,0438979	2,7838346	1,3569653
Porcentaje de varianza		0,9891	0,0108	0,6722	0,3277
Total porcentaje varianza		99,99			

Tabla 18b. Resultados del análisis de factores no rotados y con rotación varimax en instancias de la categoría 1 para corridas largas

De esa forma, el método TS esta asociado ahora al Factor 1 con cargas igual de altas a las de GH, HAS-QAP y HAS-QAP^R, y el Factor 2 esta asociado únicamente al método SA. Valores similares de las cargas para los métodos TS, GH, HAS-QAP y HAS-QAP^R en el Factor 1, los muestra fuertemente correlacionados, demostrando el porque del comportamiento parecido de sus desempeños en las instancias de esta categoría para corridas largas. La representación gráfica de la figura 14b los ubica a todos ellos casi al mismo nivel en el eje que representa al Factor 1, en el análisis de factores no rotados. En ella, el método SA se encuentra muy por debajo del resto de métodos, y cercano al eje que representa al Factor 2, de ese modo queda identificado como el peor método en instancias de la categoría 1.

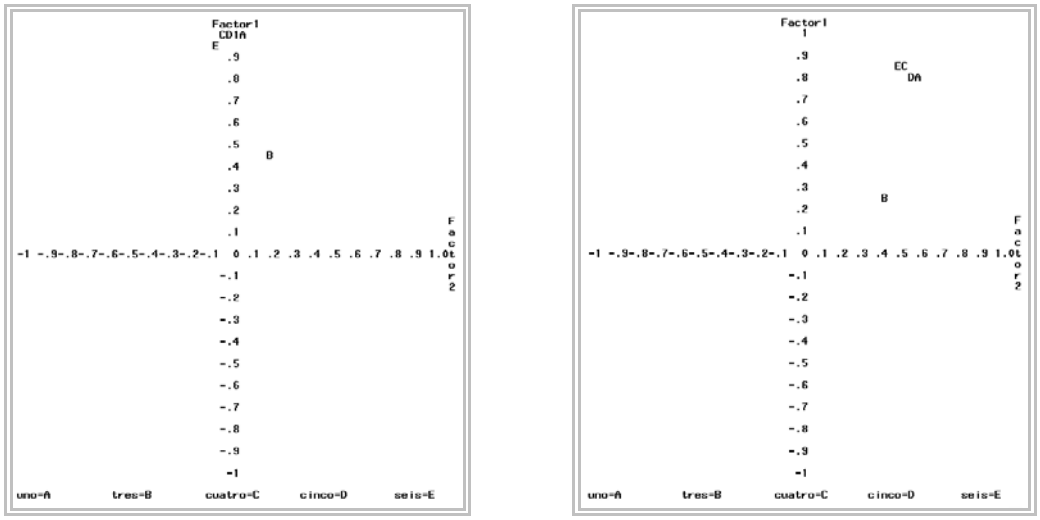


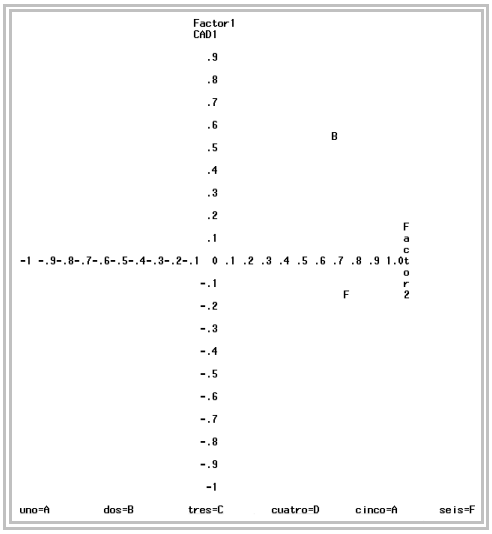
Figura 14b. Representación gráfica de los métodos dentro de los factores no rotados (izquierda) y con rotación varimax (derecha) en instancias de la categoría 1 para corridas largas

8.3.2. Análisis de factores en la categoría 2

El análisis de factores no rotados y con rotación varimax en instancias de la categoría 2 para corridas cortas retuvo sólo dos factores, el Factor1 y el Factor2. Con un total de la varianza explicada del 99,99% en corridas cortas y largas, los desempeños obtenidos por los métodos en la categoría 2 están significativamente representados por los factores 1 y 2. En la tabla 19a se observan los resultados del análisis. De acuerdo con ellos, el Factor1 esta asociado a los métodos GH, HAS-QAP, TS y SA con cargas significativamente altas, y el Factor2 esta asociado a los métodos RTS y HAS-QAP^{R} con cargas altas.

		Análisis de Factores No Rotados		Análisis de Factores con rotación varimax	
Trat	Métodos	Factor 1	Factor 2	Factor 1	Factor 2
5	HAS-QAP	0,99421	-0,05953	0,99580	0,01951
1	TS	0,98736	-0,06443	0,98936	0,01408
4	GH	0,97687	-0,03071	0,97623	0,04686
3	SA	0,97658	-0,09199	0,98080	-0,01425
6	HAS-QAP-R	-0,15634	0,76388	-0,15634	0,76388
2	RTS	0,54087	0,66944	0,54087	0,66944
Eigenvalores		4,1883	1,048	4,1685	1,0685
Porcentaje varianza		0,7998	0,2001	0,7959	0,2040
Total porcentaje varianza		99,99			

Tabla 19a. Resultados del análisis de factores no rotados y con rotación varimax en instancias de la categoría 2 para corridas cortas



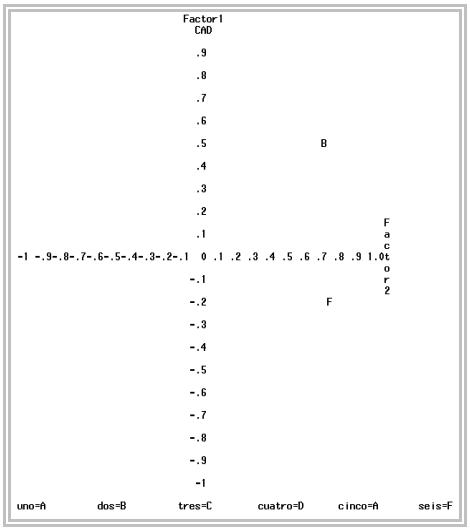


Figura 15a. Representación gráfica de los métodos dentro de los factores no rotados (izquierda) y con rotación varimax (derecha) en instancias de la categoría 2 para corridas cortas. Los tratamientos uno y cinco (TS y HAS-QAP) están representados por una única letra (letra A), al ocupar la misma posición en la gráfica

La figura 15a muestra la ubicación de los métodos dentro de los factores rotados y con rotación varimax. En ella se observa la alta correlación (cargas muy parecidas) entre los métodos GH, HAS-QAP, TS y SA, ubicados al mismo nivel en el eje que representa al Factor 1. Lo anterior explica el comportamiento parecido de sus desempeños en las instancias de la categoría 2, que se demuestra por la clasificación dada por el análisis de componentes principales en la tabla 16 para las instancias de esta categoría. El método RTS se encuentra en un punto intermedio dentro del primer cuadrante debido principalmente a que es el método con los peores desempeños en instancias de la clase b, pero con los mejores en las de la clase a comparado con el de GH, HAS-QAP, TS y SA. El método HAS-QAP^R se encuentra en un punto extremo en el cuarto cuadrante, por ser el método que mostró el mejor desempeño en las dos clases de instancias que conforman esta categoría, con desempeños muy superiores al promedio entregado por los métodos GH, HAS-QAP, TS y SA, y aún al de los entregados por RTS para la clase a.

		Análisis de Factores No Rotados		Análisis de Factores con rotación varimax	
Trat	Métodos	Factor 1	Factor 2	Factor 1	Factor 2
4	GH	0,99789	0,05243	0,99850	-0,03903
1	TS	0,98577	0,14034	0,99447	0,04961
5	HAS-QAP	0,98013	0,02443	0,97826	-0,06529
3	SA	0,96931	-0,08370	0,95759	-0,17198
2	RTS	-0,40629	0,67587	-0,34279	0,71029
6	HAS-QA-R	0,31042	0,45476	0,35070	0,42447
Eigenvalores		4,1291	0,6937	4,1004486	0,7225086
Porcentaje de varianza		0,8561	0,1438	0,8501	0,1498
Total porcentaje varianza		99,99			

Tabla 19b. Resultados del análisis de factores no rotados y con rotación varimax en instancias de la categoría 2 para corridas largas

El análisis de factores en corridas largas sirvió para hacer distinciones dentro de los métodos GH, HAS-QAP, TS y SA, y para mostrar como se aleja el método RTS de ellos al sólo conseguir mejorar sus desempeños para las instancias de la clase a con respecto a las que obtiene en corridas cortas. RTS no consiguió mejorar las soluciones que obtuvo para las instancias de la clase b en corridas largas, y por eso sus desempeños en ellas son iguales a los obtenidos en corridas cortas. La figura 15b muestra la ubicación de los métodos dentro de los factores no rotados y con rotación varimax.

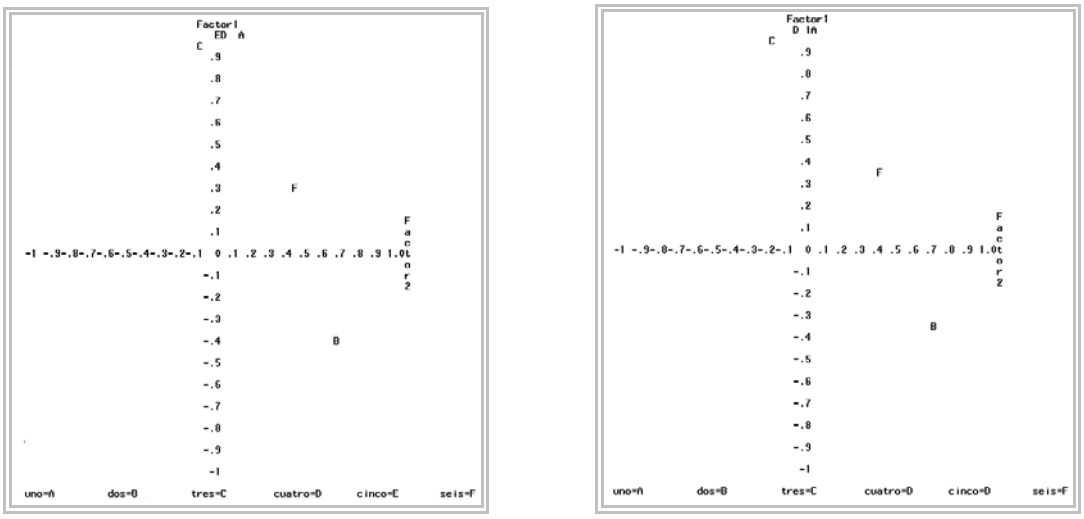


Figura 15b. Representación gráfica de los métodos dentro de los factores no rotados (izquierda) y con rotación varimax (derecha) en instancias de la categoría 2 para corridas largas

La figura 15b refleja el hecho de que dentro de los métodos GH, TS, HAS-QAP y SA, los dos primeros, GH y TS tengan los mejores desempeños en ambas clases de instancias, y HAS-QAP y SA, tengan los peores. El método RTS se ubica ahora en un punto intermedio del cuarto cuadrante, debido a que aunque sigue siendo el

método con los mejores desempeños en instancias de la clase a, frente al de GH, HAS-QAP, TS y SA, sigue teniendo los peores en instancias de la clase b. Al no lograr mejorar sus desempeños en las instancias de la clase b, los que obtiene para ellas están aún más alejados de los obtenidos por los métodos GH, TS, HAS-QAP y SA HAS-QAP^R en corridas largas. Lo anterior se identifica en la gráfica, por que RTS (representado por la letra B) toma un valor negativo en el eje que representa al Factor 1, debido a la caga significativamente negativa que tiene asociada en este factor. La ubicación del método HAS-QAP^R dentro de la gráfica esta determinado por las cargas promedio que tiene en ambos factores. Estos valores se deben a que los desempeños que entrega HAS-QAP^R para las instancias de la clase a son muy superiores a las que entregan GH, TS, HAS-QAP y SA, pero en las instancias de la clase b, están al mismo nivel.

Capítulo IX

EVALUACIÓN DEL ALGORITMO HAS-QAP^R

En esta sección se examina el comportamiento del algoritmo implementado HAS-QAP^R resolviendo las diferentes clases de instancias. Para ello se tomaron aleatoriamente instancias de cada clase en ambas categorías. Las gráficas que se muestran corresponden al comportamiento de HAS-QAP^R en corridas cortas y largas observado en las instancias seleccionadas. La tabla 23 da el listado de las instancias seleccionadas dentro de cada categoría.

Categoría 1		Categoría 2	
Clase c	Clase d	Clase a	Clase b
Bur26e	tai20b	tai25a	nug20
Els19	tai80b	tai60a	sko64
kr30a		wil50	

Tabla 20. Instancias seleccionadas para la evaluación del algoritmo HAS-QAP^R

9.1. Evaluación del algoritmo HAS-QAP^R en corridas cortas

Las gráficas HAS-QAP-R reportan la evolución de la convergencia del algoritmo hacia las buenas soluciones en función del valor de la función objetivo para cada iteración. Las iteraciones corresponden a la corrida que obtuvo la mejor solución dentro de las diez que se corren, de ahí su nombre iteraciones corrida x, donde x indica el número de ella. La línea verde (Mejor Solución Iter) representa el comportamiento de la mejor solución encontrada en cada iteración y la morada (Comp.MG) el de la mejor solución global. La línea salteada en azul (Promedio Iter) representa a la solución promedio de las diez iteraciones. La iteración 0 esta asociada a la mejor solución global obtenida en la inicialización del algoritmo y que se obtiene a partir de soluciones aleatorias optimizadas con búsqueda local. Los puntos en

amarillo (Mejoras) indican las iteraciones en los que se obtuvo una mejora de la mejor solución global encontrada hasta esa iteración. Los puntos en rojo (Diversificación) representan las iteraciones en donde se hace necesario reinicilizar el algoritmo en la siguiente iteración.

A continuación se observan las gráficas HAS-QAP-R para las instancias de la categoría 1 en corridas cortas (10 iteraciones). De acuerdo con ellas el algoritmo HAS-QAP^R encuentra mejoras significativas de la mejor solución global en cada iteración, dadas por puntos amarillos casi consecutivos y por tanto una mejora de las soluciones conseguidas en cada iteración que reflejan la intensificación de la búsqueda. Los intervalos correspondientes a no mejoras (iteraciones sin puntos amarillos –Mejoras-) de la mejor solución global permiten identificar las iteraciones en los que el algoritmo pone a prueba su mecanismo de exploración mediante la aplicación de la regla de exploración en los R intercambios. De esta forma demuestra que puede obtener soluciones no sólo diferentes a las que venía encontrando sino de mejor calidad. Esta situación se ve con más claridad en la figura 18c dada para la instancia kr30a. Dado que en corridas cortas HAS-QAP^R no emplea su mecanismo de diversificación (no aparición de puntos rojos), que es uno de los que más lo caracteriza, éste, sin embargo, puede ser reemplazado por la regla de explotación en la exploración de nuevas y mejores regiones de búsqueda que lo hacen efectivo en ambas categorías de instancias al arrojar buenas soluciones en corto tiempo computacional con tan pocas iteraciones de prueba.

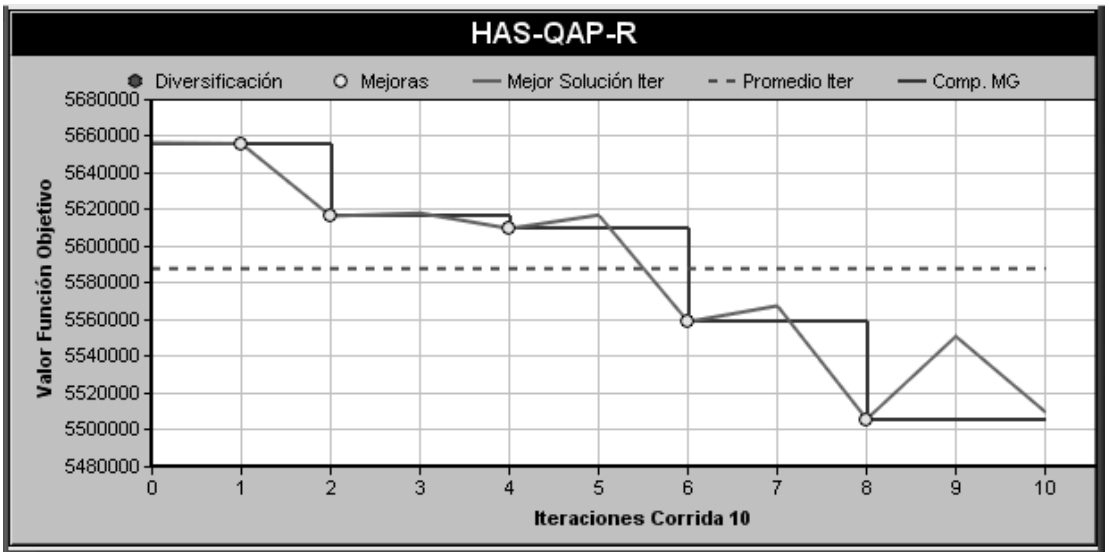


Figura 16a. Gráfica HAS-QAP-R para la instancia bur26e en corridas cortas

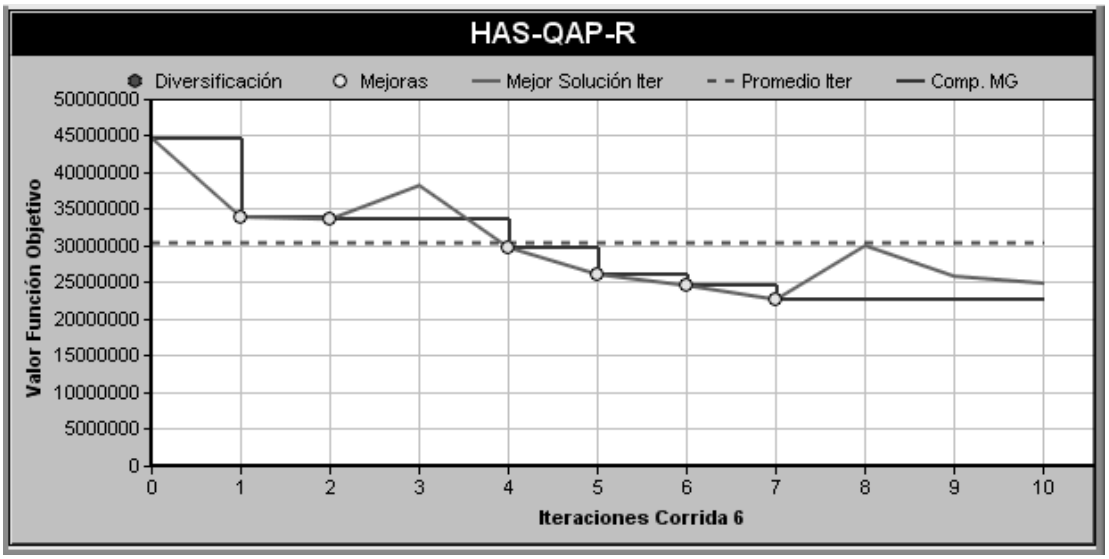


Figura 16b. Gráfica HAS-QAP-R para la instancia els19 en corridas cortas

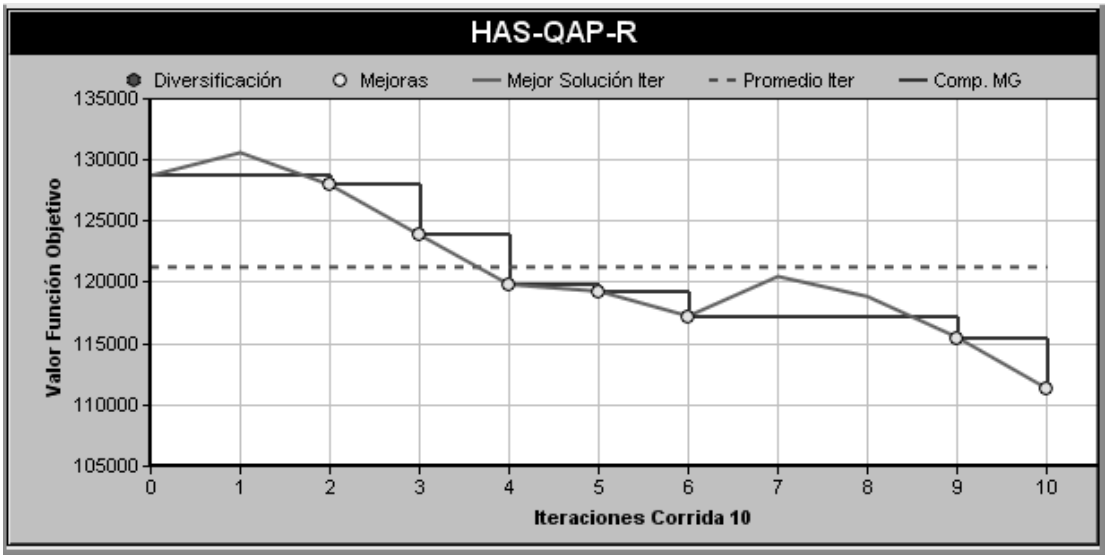


Figura 16c. Gráfica HAS-QAP-R para la instancia kr30a en corridas cortas

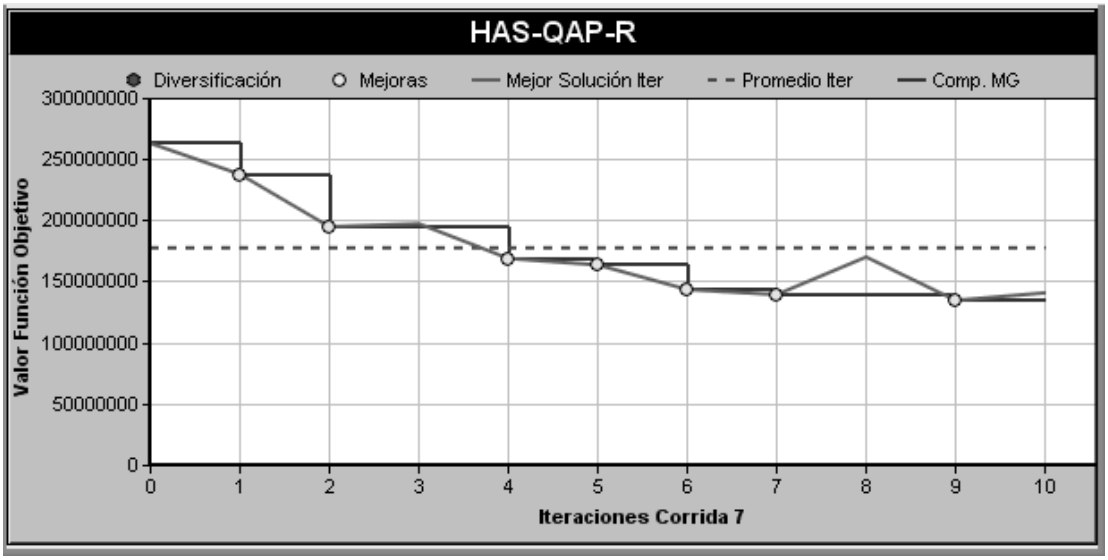


Figura 16d. Gráfica HAS-QAP-R para la instancia tai20b en corridas cortas

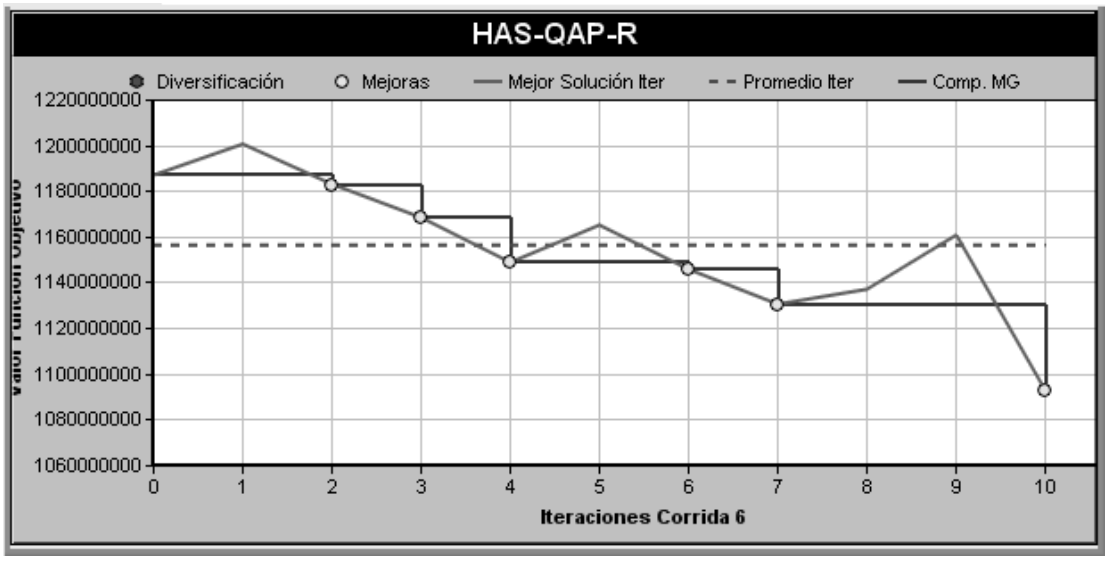


Figura 16e. Gráfica HAS-QAP-R para la instancia tai80b en corridas cortas

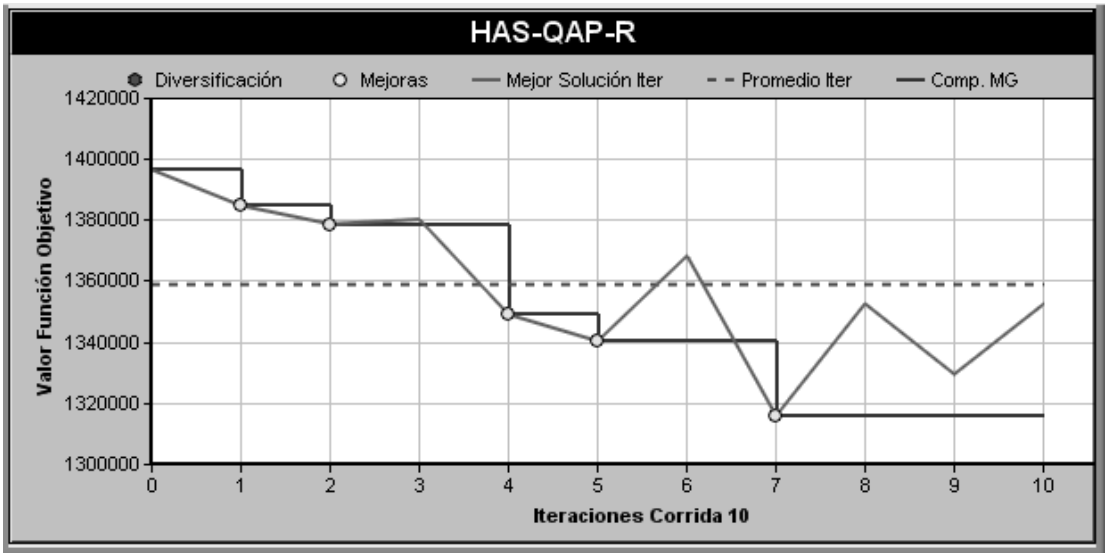


Figura 17a. Gráfica HAS-QAP-R para la instancia tai25a en corridas cortas

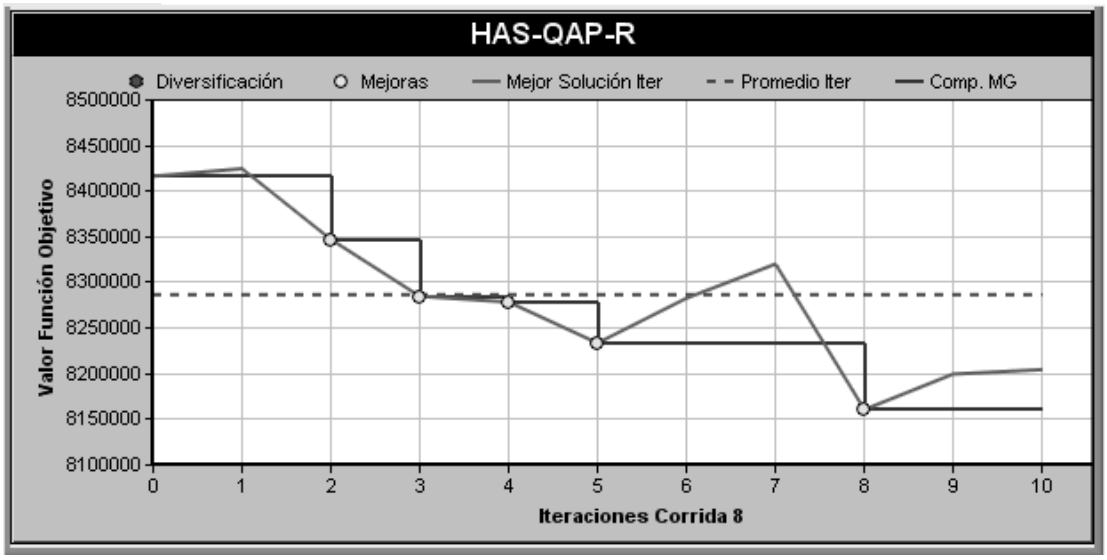


Figura 17b. Gráfica HAS-QAP-R para la instancia tai60a en corridas cortas

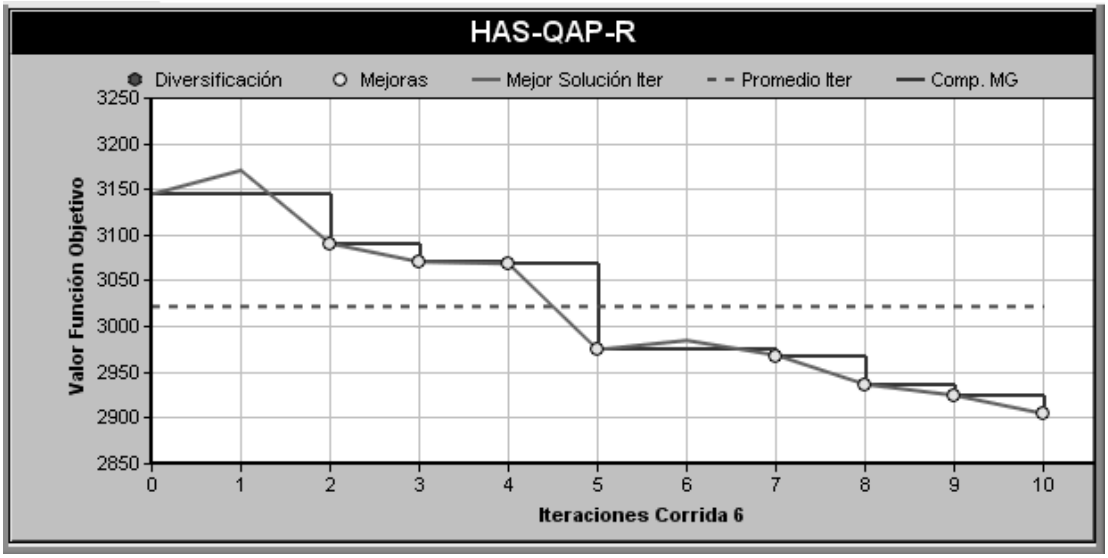


Figura 17c. Gráfica HAS-QAP-R para la instancia nug20 en corridas cortas

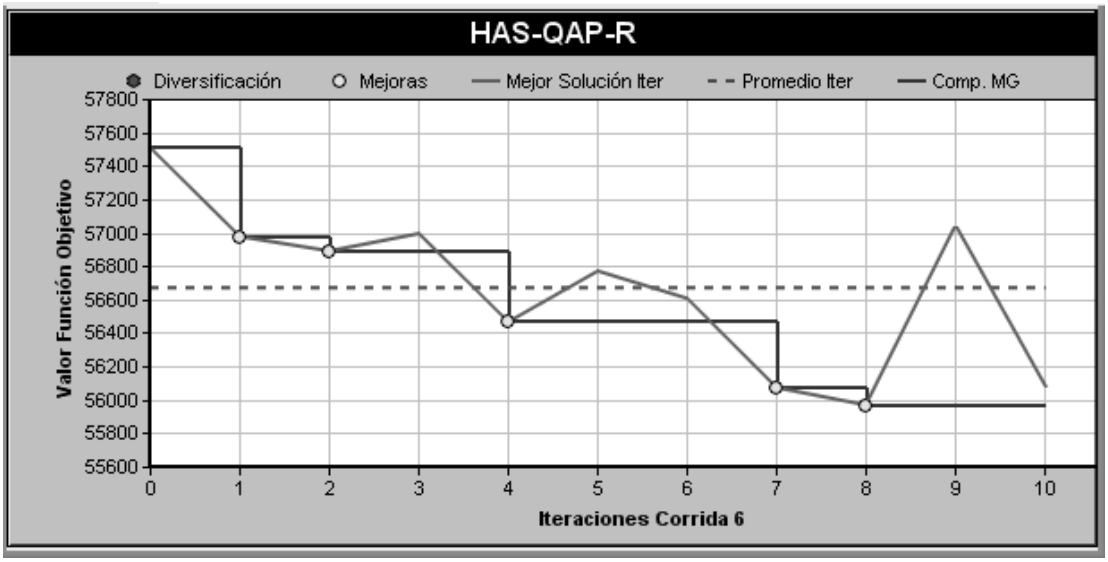


Figura 17d. Gráfica HAS-QAP-R para la instancia sko64 en corridas cortas

9.2. Evaluación del algoritmo HAS-QAP^R en corridas largas

El comportamiento del algoritmo HAS-QAP^R en corridas largas se divide en dos etapas, la primera constituye una fase de búsqueda que parte de soluciones generadas aleatoriamente de muy baja calidad, que el algoritmo iteratívamente mejora con intervalos reducidos entre Mejoras. Por esa razón el número de ellas es mucho mayor en esta etapa comparado con el de la segunda. Esta etapa entrega a la segunda una solución más definida que permite identificar otras regiones desde donde seguir explorando para optimizarla. En la segunda etapa las mejoras no son tan significativas como en la primera y por eso en las gráficas se pueden ver al mismo nivel. Así mismo, para ciertas instancias, los intervalos en los que se consiguen son considerablemente mayores a los de la primera. La funcionalidad de la diversificación como mecanismo de exploración se observa en la gráfica con soluciones de menor calidad a las obtenidas hasta ese punto en las próximas iteraciones, hasta que llega a una en la que obtiene una Mejora (punto amarillo). En la instancia kr30a por ejemplo la mejora fue significativa con respecto a la anterior después de una diversificación. Para algunas instancias como la tai80b, la

diversificación no logra activarse porque el lapso de iteraciones determinadas para ello es superior al máximo intervalo de ellas en las que el algoritmo obtiene una mejora. De acuerdo con esto, el algoritmo trabaja bien con sólo el mecanismo de intercambios, pero indica que con un mayor número de iteraciones se puede dar paso a optimizaciones que sólo podrían obtenerse con la diversificación.

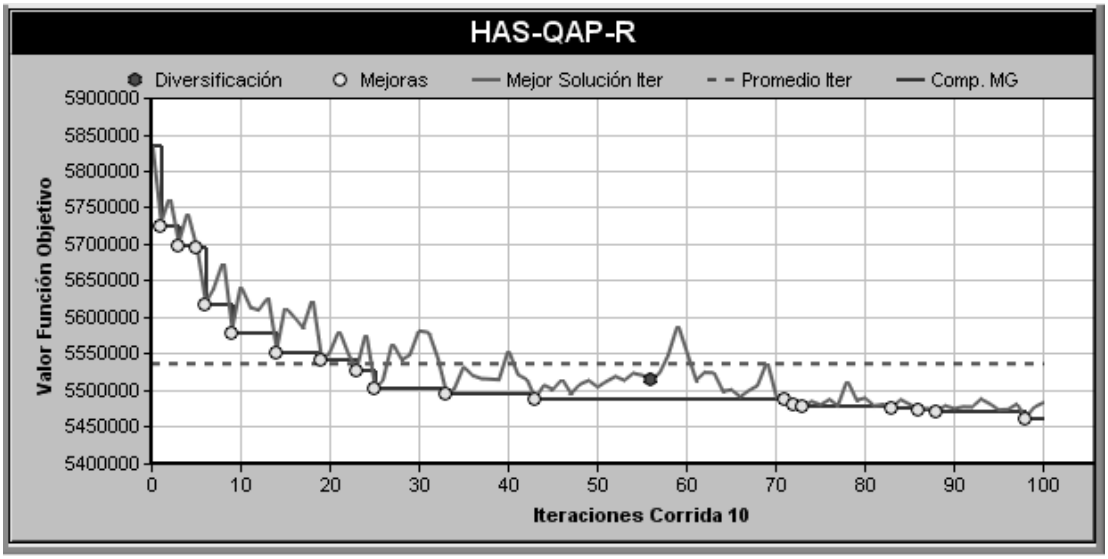


Figura 18a. Gráfica HAS-QAP-R para la instancia bur26e en corridas largas

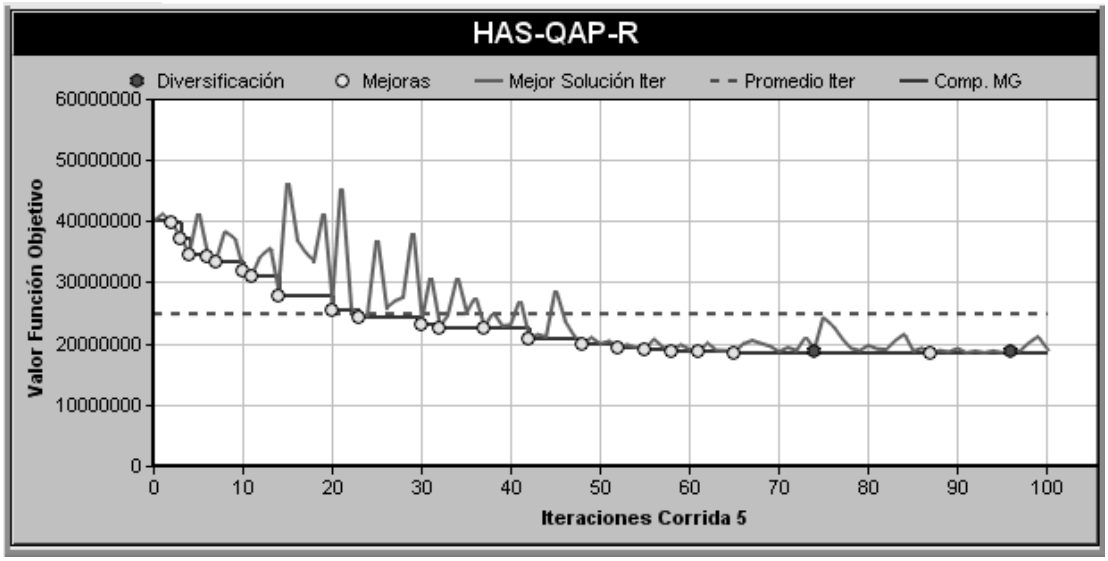


Figura 18b. Gráfica HAS-QAP-R para la instancia els19 en corridas largas

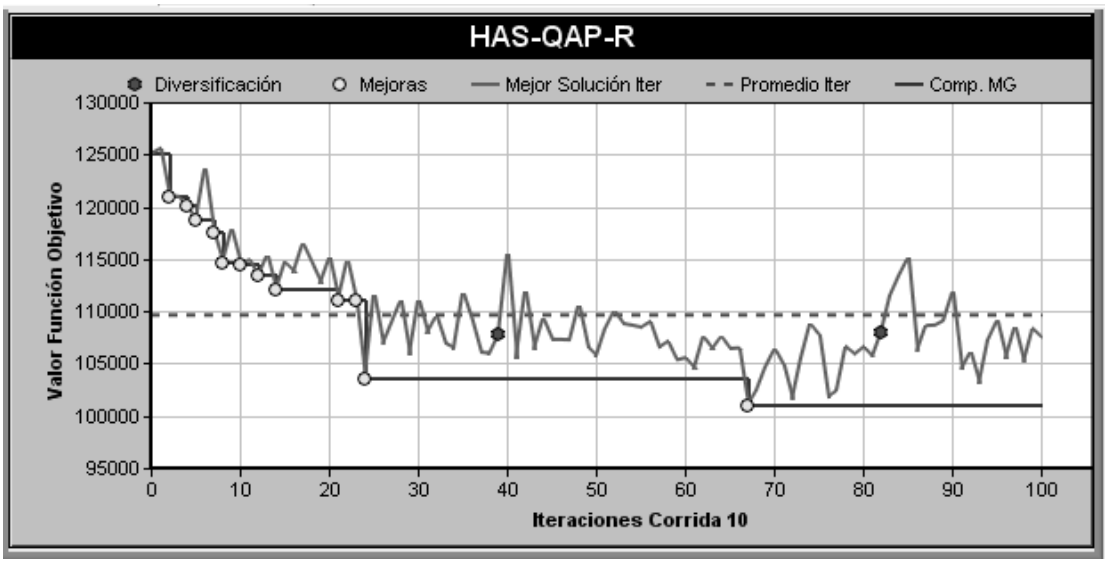


Figura 18c. Gráfica HAS-QAP-R para la instancia kr30a en corridas largas

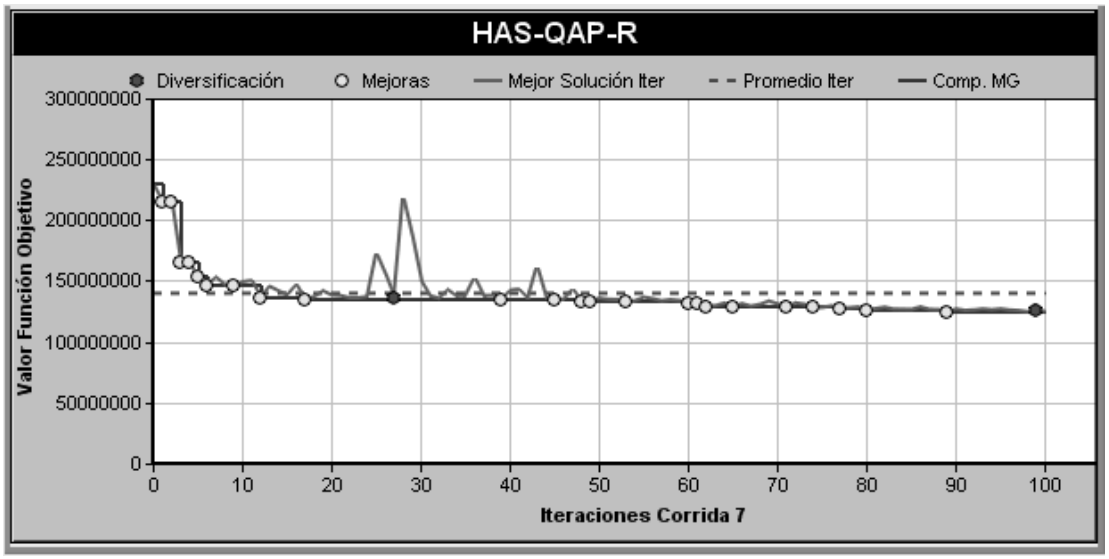


Figura 18d. Gráfica HAS-QAP-R para la instancia tai20b en corridas largas

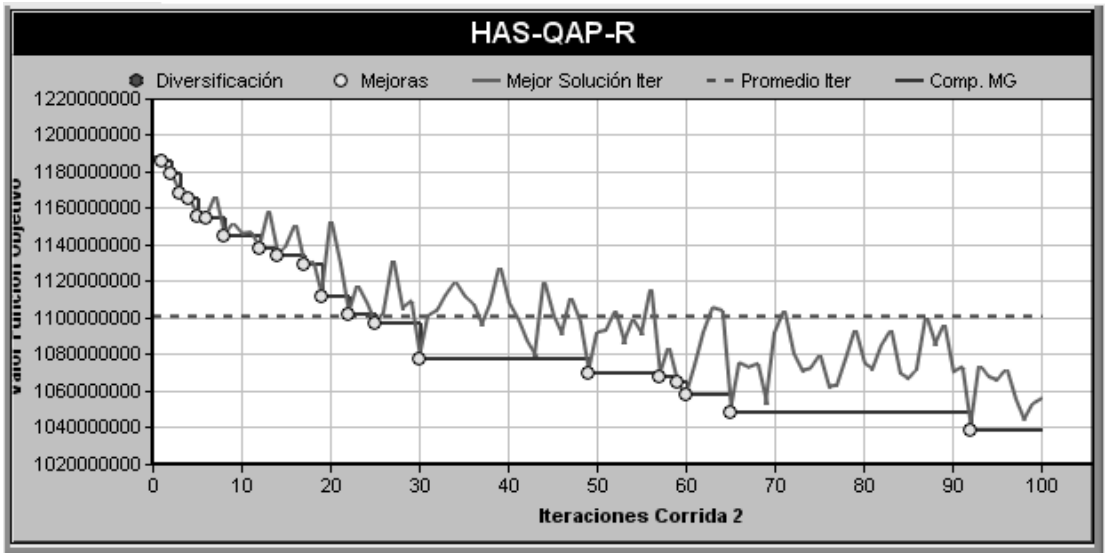


Figura 18e. Gráfica HAS-QAP-R para la instancia tai80b en corridas largas

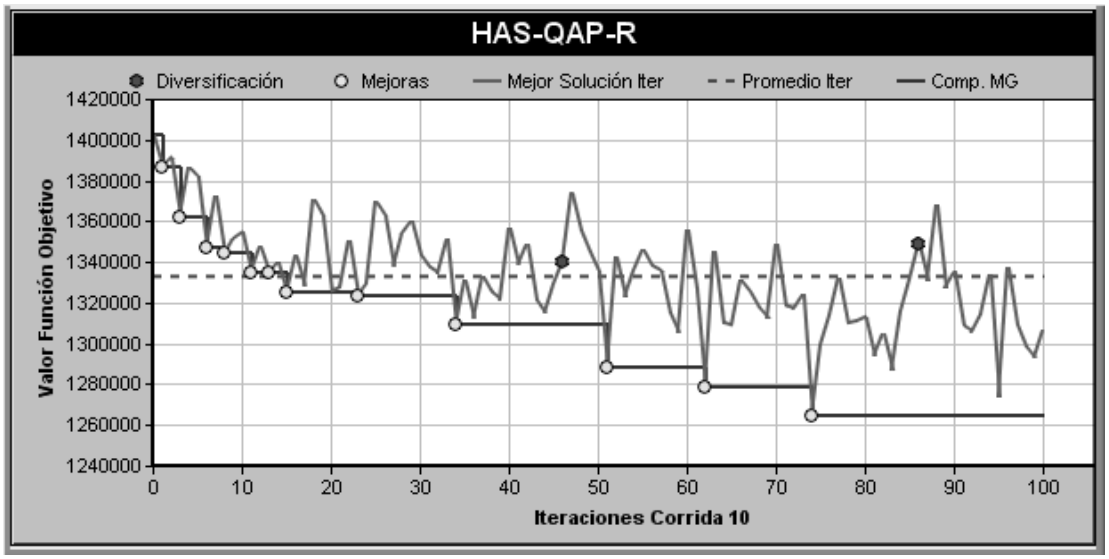


Figura 19a. Gráfica HAS-QAP-R para la instancia tai25a en corridas largas

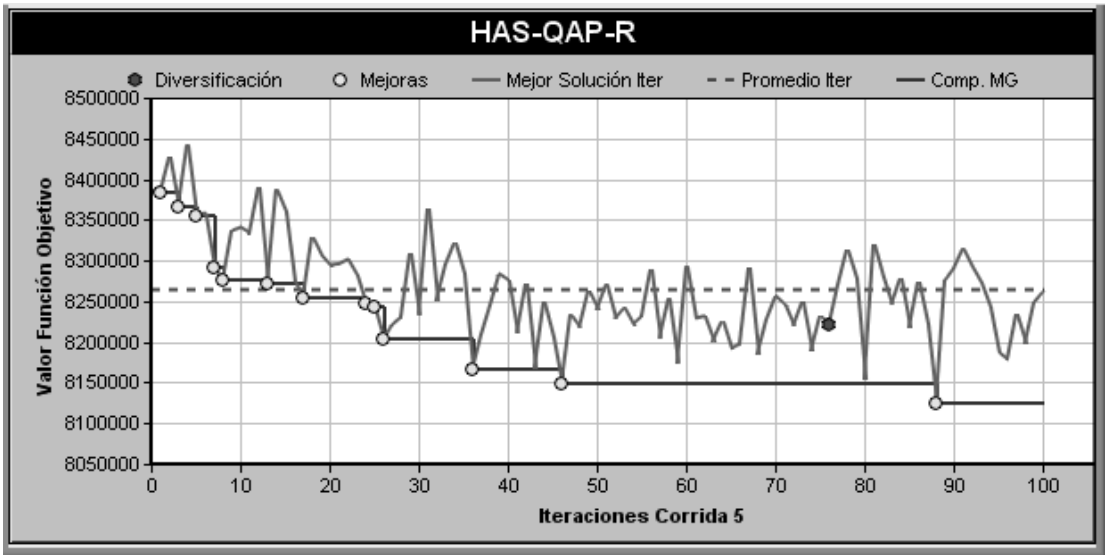


Figura 19b. Gráfica HAS-QAP-R para la instancia tai60a en corridas largas

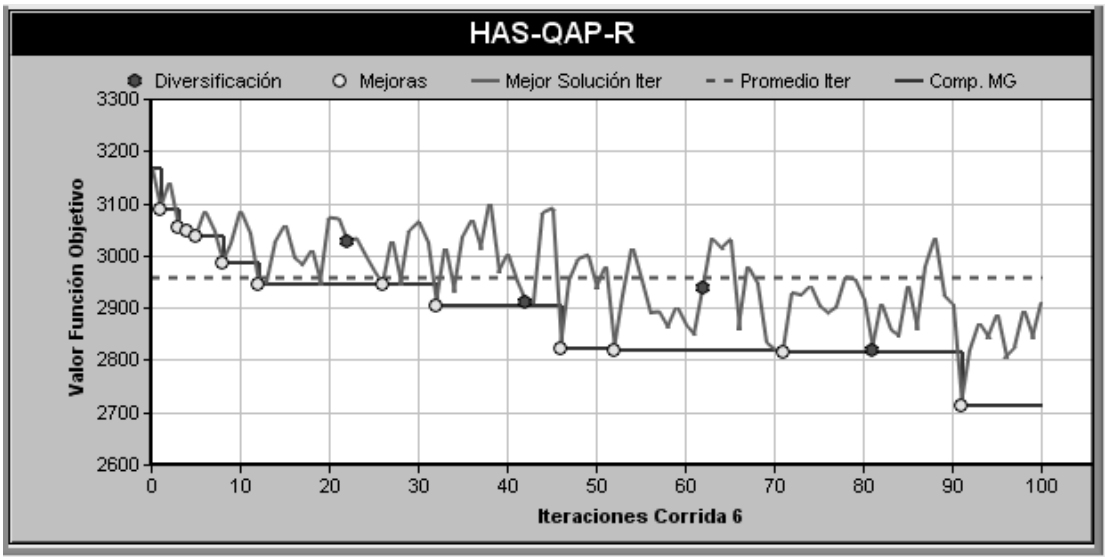


Figura 19c. Gráfica HAS-QAP-R para la instancia nug20 en corridas largas

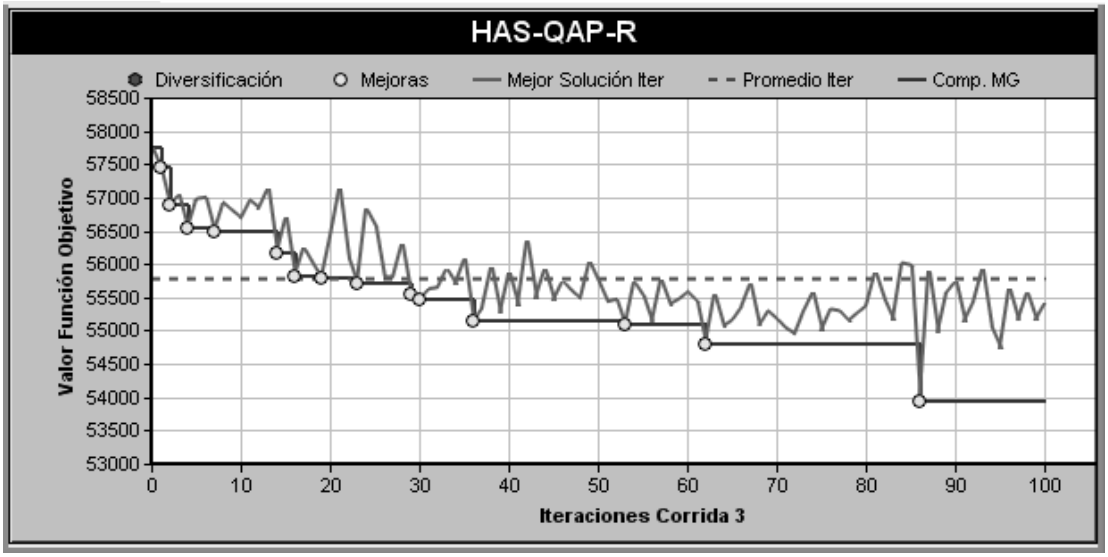


Figura 19d. Gráfica HAS-QAP-R para la instancia sko64 en corridas largas

9.3. Gráficas del comportamiento de las soluciones de HAS-QAP^R

Las gráficas que se observan a continuación corresponden a los resultados que arrojó el algoritmo HAS-QAP^R para las instancias seleccionadas en la sección 7.6 en corridas largas con los parámetros seleccionados en la sección 7.1. Los resultados comprenden la solución promedio, la mejor, y la peor solución encontrada en cada una de las diez corridas.

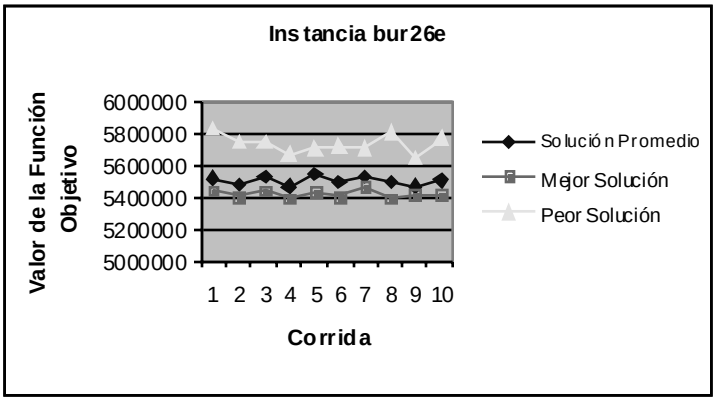


Figura 20. Comportamiento de los resultados instancia bur26e

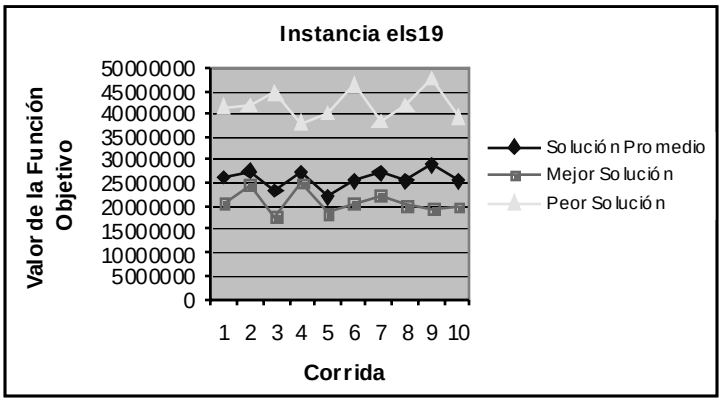


Figura 21. Comportamiento de los resultados instancia els19

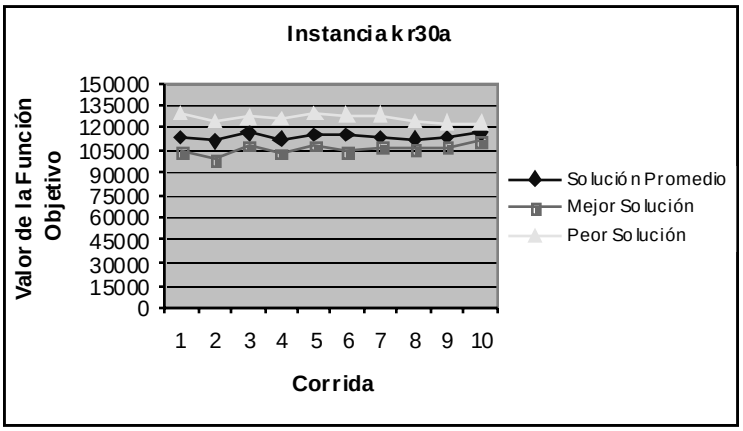


Figura 22. Comportamiento de los resultados instancia kr30a

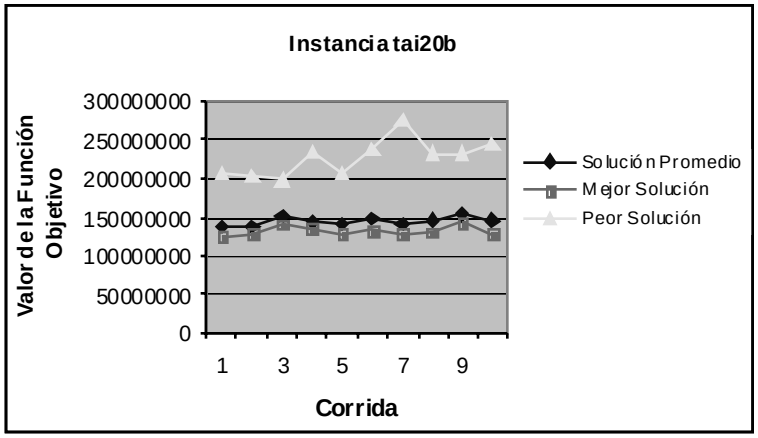


Figura 23. Comportamiento de los resultados instancia tai20b

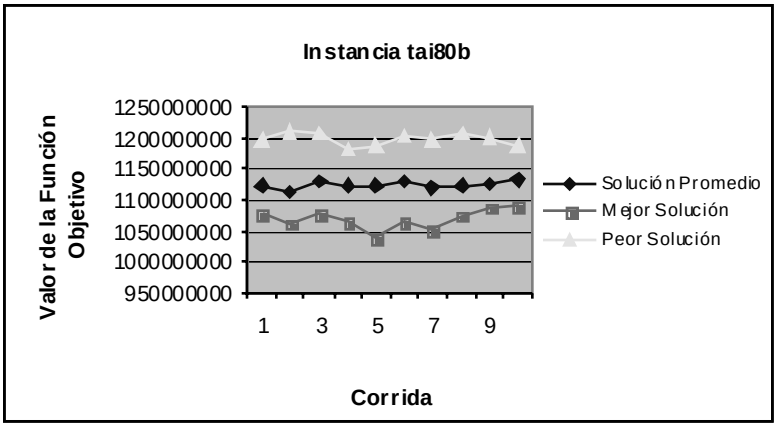


Figura 24. Comportamiento de los resultados instancia tai80b

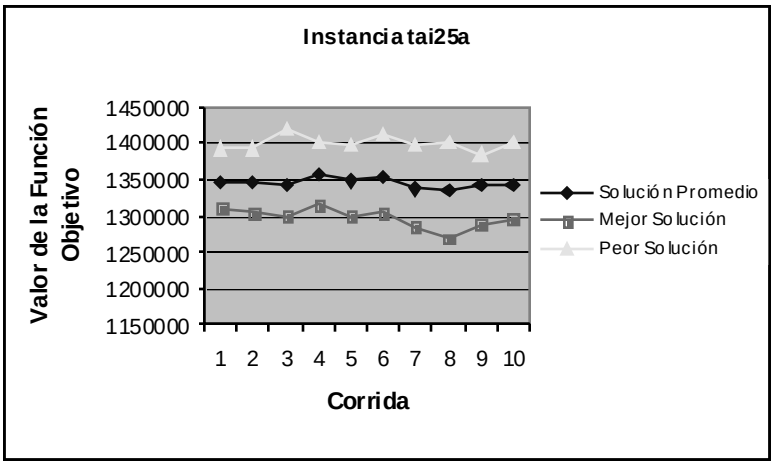


Figura 25. Comportamiento de los resultados instancia tai25a

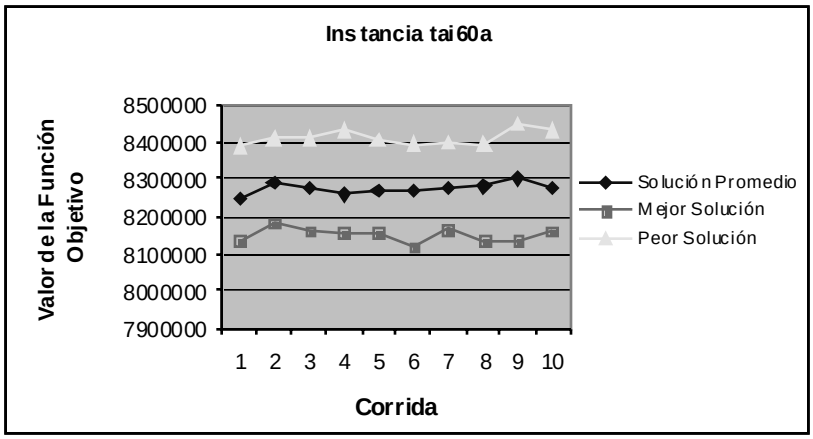


Figura 26. Comportamiento de los resultados instancia tai60a

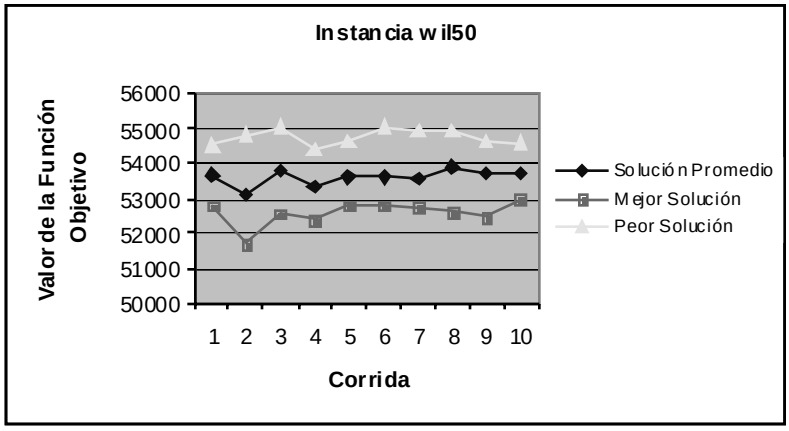


Figura 27. Comportamiento de los resultados instancia wil50

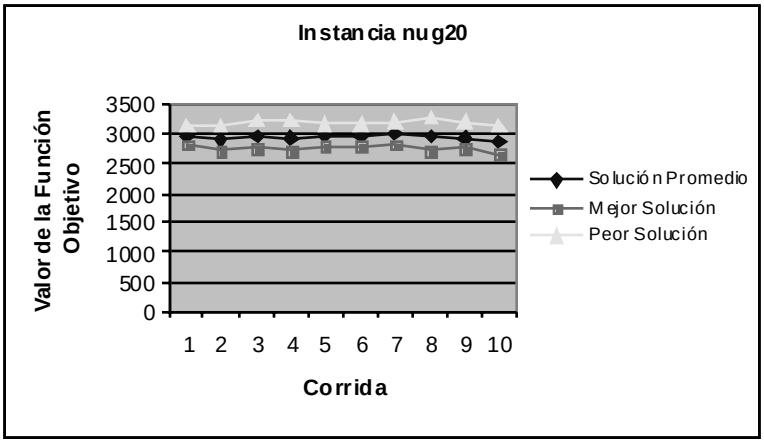


Figura 28. Comportamiento de los resultados instancia nug20

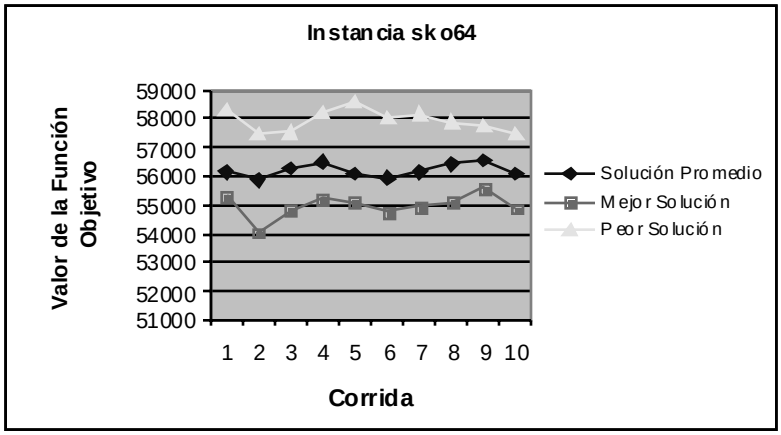


Figura 29. Comportamiento de los resultados instancia sko64

Capítulo X

DISEÑO FACTORIAL FRACCIONAL

En el diseño factorial el término factores se refiere a los componentes o mecanismos de alto nivel que caracterizan a la metaheurística, y los niveles, a los valores que un determinado factor puede tomar. El propósito de éste es determinar si los factores seleccionados son realmente significativos, es decir, si hay un cambio significativo en los promedios obtenidos por sus niveles; y determinar si las interacciones entre los factores son significativas, es decir, si hay un cambio significativo en los promedios obtenidos por las combinaciones de sus niveles. La variable respuesta corresponde a una solución obtenida por el algoritmo en corridas largas.

Categoría 1		Categoría 2	
Clase c	Clase d	Clase a	Clase b
Bur26e	kr30a	Tai25a	nug20
chr25	tai20b		sko42

Tabla 21. Instancias seleccionadas para el diseño factorial fraccional

		NIVELES		
N°	FACTOR	1	2	3
1	ρ	0,1	0,5	0,9
2	S	$n/2$	$n/6$	$>> n$
3	q	0,1	0,9	1

Tabla 22. Valores de los niveles para los tres factores

En este trabajo se tomaron en cuenta tres factores con tres niveles cada uno dados en la tabla 22, por lo que se habla de un diseño factorial 3^3, sin embargo, resulta muy complejo ejecutar las 3^3 corridas del diseño completo (medirla en la totalidad de las

instancias), por esa razón, se recurrió a un diseño factorial fraccional que recurre a diseños corriendo sólo una fracción de todo el experimento que corresponde al grupo de instancias seleccionadas para ello (ver tabla 21). El número de tratamientos es de veintisiete, formados a partir de las combinaciones de los niveles de los tres factores (ver tabla 23). La variable corrida (réplica del algoritmo) al no ser objetivo del estudio es tratada como el factor bloque con diez niveles (diez réplicas). El procedimiento utilizado para efectuar el diseño fue el Proc GLM del programa SAS. Más adelante se presentan las conclusiones generales del diseño.

	Combinación Niveles Factores		
Trat	ρ	S	q
1	1	1	1
2	1	1	2
3	1	1	3
4	1	2	1
5	1	2	2
6	1	2	3
7	1	3	1
8	1	3	2
9	1	3	3
10	2	1	1
11	2	1	2
12	2	1	3
13	2	2	1
14	2	2	2
15	2	2	3
16	2	3	1
17	2	3	2
18	2	3	3
19	3	1	1
20	3	1	2
21	3	1	3
22	3	2	1
23	3	2	2
24	3	2	3
25	3	3	1
26	3	3	2
27	3	3	3

Tabla 23. Tratamientos formados por las combinaciones de los niveles de los tres factores

Las soluciones obtenidas con cada uno de los veintisiete tratamientos para las instancias seleccionadas se muestran a continuación.

	Corrida									
Trat	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	5507059	5449208	5475312	5455020	5469705	5452675	5474242	5488203	5467370	5475289
2	5459381	5478533	5485488	5498066	5485657	5464733	5477101	5444139	5476116	5484800
3	5483937	5510148	5448773	5489559	5463836	5486607	5471073	5472136	5452157	5544535
4	5505813	5485703	5464208	5483486	5498874	5495583	5488032	5474566	5493279	5471979
5	5477708	5481134	5451607	5473835	5455585	5490367	5479967	5481762	5500915	5480575
6	5479907	5505555	5481357	5476536	5483501	5454759	5478315	5463114	5473951	5486650
7	5463691	5485051	5482801	5476381	5491951	5457252	5483875	5465965	5475102	5461445
8	5483839	5493677	5481392	5469966	5503425	5471757	5484934	5455352	5457088	5491905
9	5459810	5454606	5460169	5484347	5493215	5456963	5441016	5484291	5491397	5449187
10	5479385	5475823	5451625	5475598	5474679	5492560	5469230	5475870	5455870	5488435
11	5445901	5497387	5446484	5463546	5469582	5479848	5482262	5465419	5481954	5481695
12	5481665	5508829	5465477	5478287	5483746	5477612	5503256	5490603	5491641	5458372
13	5473233	5504933	5473843	5495880	5457395	5461236	5465237	5493269	5467414	5486947
14	5464873	5485154	5494090	5495016	5463673	5462744	5495898	5497981	5482128	5483675
15	5459770	5491632	5466239	5465433	5474611	5477744	5471693	5472039	5477312	5468750
16	5452701	5457467	5470504	5486093	5452127	5456128	5490201	5465051	5482984	5459041
17	5465025	5472120	5476897	5482055	5480137	5469372	5467944	5502489	5493126	5491172
18	5462242	5463262	5514072	5450112	5467097	5473130	5453862	5460788	5482162	5463183
19	5471894	5465587	5484773	5478991	5471700	5461336	5465028	5463599	5504652	5501711
20	5454425	5467988	5464702	5458106	5458601	5482317	5480820	5499979	5466037	5447672
21	5462401	5481927	5489332	5476969	5449091	5460070	5473976	5480306	5445688	5471849
22	5455546	5480380	5461702	5478928	5473386	5458959	5477264	5480408	5482877	5449350
23	5482432	5471913	54826554	5505657	5503032	5471903	5486935	5506232	5468826	5497450
24	5482240	5457313	5494703	5492274	5478942	5473497	5464886	5469706	5460324	5475259
25	5476857	5479309	5463571	5479835	5462709	5454422	5449001	5479656	5461979	5451964
26	5449378	5460649	5506101	5464483	5460076	5451988	5464068	5476419	5460622	5481926
27	5449290	5464297	5461853	5457152	5468108	5466645	5483530	5458803	5460984	5460282

Tabla 24. Respuestas obtenidas por los veintisiete tratamientos en diez corridas para la instancia bur26e

	Corrida									
Trat	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	8652	9032	9482	9400	8416	8498	9458	8000	9358	8800
2	9284	8998	7810	8810	9434	9732	7488	9386	9784	9730
3	7498	9594	8178	9852	6924	10030	9638	9074	8788	9204
4	10078	10606	8788	9382	9682	8854	9534	9780	10726	9084
5	7652	9692	8330	9080	8486	9478	9074	9812	9352	7788
6	8128	8806	9068	8500	8916	7666	6648	8472	8420	9130
7	7134	8360	7774	9124	8790	8486	7736	9058	9910	8942
8	9192	9268	8276	8472	9372	8802	9084	7540	7662	8966
9	7276	8502	8320	9078	8406	11304	9288	7778	9252	8128
10	9038	9276	9286	8576	8896	9410	7812	9486	10184	8714
11	8072	8380	9156	10386	8380	9336	8474	10250	8048	9838

12	9056	8518	9602	8780	8204	9566	10292	9888	7384	9330
13	9362	10076	8434	8094	11204	8194	9508	9276	10414	8946
14	8900	7950	7698	8982	7788	7968	7430	8524	10444	7698
15	8590	9906	9018	9320	9604	9668	8306	8908	9568	9806
16	9588	8802	8676	8222	9728	8590	8494	7448	9812	9178
17	8966	7936	8060	9722	9772	8144	10490	9366	8372	6746
18	9052	9296	8540	9374	10494	8528	9656	8554	8552	8502
19	8832	9000	8472	10124	9628	8674	8386	9034	8448	9234
20	10896	10252	8882	9378	8740	9256	10046	8874	8506	8240
21	8608	9054	8648	10522	7308	9966	11136	8074	8602	8616
22	9522	9550	10748	9406	9524	9460	7966	9586	8772	9866
23	8870	8142	9804	8190	9142	9598	7560	7448	8046	7710
24	9728	8502	8904	10210	9502	9024	8802	8912	8988	11050
25	6254	6970	7672	8410	9888	8990	8038	8436	7226	8728
26	8264	7106	6268	7524	7830	9458	9610	7136	7590	9612
27	10002	9526	8816	8932	8416	10426	8658	8630	8724	10092

Tabla 25. Respuestas obtenidas por los veintisiete tratamientos en diez corridas para la instancia chr25

	Corrida									
Trat	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	110360	103575	110205	103420	105150	110385	105890	107540	102905	100800
2	105140	108575	101900	102535	108415	109580	100635	106075	108280	105220
3	105120	102170	104220	104430	107470	104855	105650	107605	108450	102045
4	111120	101700	109085	108210	104775	107900	102100	109725	105070	103780
5	102085	103670	106530	108985	107435	105850	103405	105470	109395	103385
6	105485	105340	107835	105815	102275	106135	105805	107395	108125	110165
7	106780	107755	109110	107660	105200	106445	102265	102040	105190	104515
8	106940	104645	104840	107715	107440	109385	107280	94865	103320	105895
9	101700	104475	101310	104645	105905	100375	105225	104255	104675	98545
10	100220	101375	102985	105710	106005	107840	99475	95890	108775	98010
11	106730	105860	108130	100295	105425	100915	104635	101700	108525	104350
12	103760	96345	111230	106505	103520	100475	104375	106645	101185	109465
13	103090	110375	105400	107995	103680	106905	106480	105885	108450	109620
14	100990	102420	104245	109875	102230	106325	109635	105620	106100	105320
15	102725	106560	101570	105535	107065	103565	109125	107745	110685	108745
16	103335	98990	104910	106140	107370	105065	105265	104440	106240	108925
17	105865	110885	102675	105990	105440	99615	108315	105305	105990	108755
18	107025	108780	106175	110360	107010	104100	107955	106335	107690	108190
19	106375	105875	107800	108350	107835	103390	99720	108570	104730	102075
20	108100	106205	109145	102345	105650	103140	104945	105655	109055	105790
21	104160	99645	106580	106595	106595	105580	105460	102960	108885	110090
22	107875	103335	108010	105490	108240	105415	99870	100285	108150	107085
23	108495	107995	104450	106835	101205	103445	104615	108695	103675	108725
24	103630	106615	109545	101945	100920	106755	106025	108015	97450	110895
25	99970	108925	105955	104045	104520	106700	105550	105150	102570	107010
26	103600	99135	99410	106840	109875	106375	106485	104080	102390	106230
27	103905	100795	103900	102865	107765	104640	106455	104905	104550	96810

Tabla 26. Respuestas obtenidas por los veintisiete tratamientos en diez corridas para la instancia kr30a

	Corrida									
Trat	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	127897744	128174105	127416692	127091117	127365967	126273224	137850442	128455567	132464005	123904706
2	125547656	126307174	126346492	126619224	135615206	138902153	137544190	137427241	154563511	142228562
3	125276795	129980258	127315600	128196778	125269450	138718225	139437183	154206787	138949657	126332742
4	127076446	126989637	128219274	129376582	127959294	130262955	141544468	129877921	140563652	127445671
5	128810143	126350243	129670636	138684712	138604949	126211095	129795398	130295638	129828443	128017381
6	126129450	696411397	140442791	140154134	161971096	127966284	127824223	136568386	137792370	142049378
7	125438455	127175419	138532037	130528908	138777922	123622734	137417528	126433121	137373469	124558204
8	123267521	137804355	139476957	128356370	131686200	138501003	128959129	128845077	136044391	137342903
9	124787496	127208547	127985532	125264412	139124122	126035921	135735465	126447571	140185740	136789734
10	124433006	125714279	126941849	138476556	126583074	126027498	124636614	123867448	125892036	126648193
11	129114743	129293245	127098198	126307405	138696794	136696794	127995274	127588391	128720029	140349172
12	127233171	137020806	126740494	129267556	128447335	126875194	125240837	126404579	131086481	126406796
13	125774692	126959648	125606701	160188093	126870826	141272538	129370875	129256112	128697176	129569182
14	125874139	124327795	125925103	140379841	128843221	125575840	126488745	137203319	126646815	129895294
15	126462281	170336117	130404111	127465127	128619412	130311727	136540556	127966564	125051276	129679848
16	126036956	124269702	135988035	129348889	127571318	140141312	136888286	137027332	136639305	127851690
17	126697663	125839393	127273515	127469215	129587455	127372969	125827064	137790128	127848574	128215193
18	126071782	124913161	140788483	127227978	126114607	141300659	139305439	136665215	136884420	126959608
19	128725152	138896812	137294806	123465805	124980336	130017761	129664276	128619327	126521727	128671396
20	128522993	124569768	137918867	161444095	136969716	129121202	137022671	137404456	137194567	127722971
21	125268501	126330682	137332836	127328200	130730921	126854200	127532244	124684663	130811217	137784908
22	126336247	137488557	125030046	125073216	126265682	127177793	129636075	138932140	141642992	141625062
23	126791448	128752967	127565323	141052301	127717448	140456036	130621300	125306192	129489056	129204613
24	127673375	127416009	131874377	130889333	125412947	132342774	128406799	129756189	136416606	128695827
25	126339555	139158677	136709866	138972556	138193352	140384084	125568722	124719756	125560825	138801071
26	124378658	125392517	134962925	138882128	156544600	126817394	139056836	124533693	137106004	136398031
27	124619344	124134171	131663662	142802860	126573488	136684731	136555148	125798394	136824878	123868512

Tabla 27. Respuestas obtenidas por los veintisiete tratamientos en diez corridas para la instancia tai20b

	Corrida									
Trat	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	2780	2836	2756	2714	2822	2770	2700	2744	2734	2764
2	2642	2788	2750	2742	2704	2750	2768	2782	2794	2836
3	2734	2760	2792	2794	2754	2656	2754	2738	2744	2804
4	2770	2744	2730	2830	2790	2790	2852	2814	2784	2748
5	2774	2786	2772	2754	2764	2682	2752	2794	2780	2686
6	2802	2750	2820	2736	2798	2764	2788	2784	2776	2790
7	2746	2820	2730	2734	2712	2738	2788	2712	2762	2794
8	2702	2796	2652	2820	2864	2776	2680	2858	2728	2706
9	2768	2736	2706	2708	2704	2800	2768	2748	2710	2762
10	2756	2730	2794	2832	2766	2742	2794	2722	2824	2788
11	2762	2714	2782	2736	2714	2696	2752	2786	2906	2786
12	2770	2756	2736	2766	2788	2700	2802	2734	2734	2736

13	2704	2818	2756	2794	2796	2800	2786	2780	2728	2768
14	2776	2790	2746	2756	2804	2772	2704	2798	2762	2764
15	2820	2832	2738	2704	2774	2768	2740	2736	2740	2776
16	2706	2682	2758	2652	2758	2670	2768	2666	2684	2800
17	2736	2816	2724	2698	2724	2788	2806	2758	2880	2692
18	2778	2732	2808	2714	2790	2712	2668	2776	2764	2784
19	2808	2826	2744	2722	2754	2694	2804	2732	2788	2760
20	2726	2860	2850	2718	2668	2700	2778	2822	2702	2778
21	2716	2760	2712	2804	2744	2688	2770	2762	2762	2722
22	2746	2724	2744	2726	2744	2800	2758	2810	2768	2744
23	2726	2760	2738	2794	2750	2686	2762	2782	2804	2680
24	2844	2724	2804	2806	2848	2860	2744	2690	2674	2762
25	2762	2746	2774	2768	2708	2732	2732	2814	2742	2692
26	2752	2814	2848	2676	2748	2686	2728	2778	2714	2732
27	2716	2730	2826	2720	2748	2724	2758	2712	2786	2728

Tabla 28. Respuestas obtenidas por los veintisiete tratamientos en diez corridas para la instancia nug20

	Corrida									
Trat	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	17982	18420	18262	17908	17834	18138	18186	17930	18094	17808
2	18090	17880	18226	18130	18214	18282	18214	18052	18224	17990
3	18182	18622	18312	18250	18340	18108	18054	18340	18020	17650
4	18220	18088	18234	18240	18086	18268	18134	17904	18286	18280
5	18080	18176	18350	18276	18346	17884	18234	17656	18232	18492
6	18056	18124	18414	18264	18226	18130	18042	18218	17998	18372
7	18094	18320	17924	18266	17994	18108	17906	18136	17926	17942
8	18432	17720	17840	18030	17964	18118	18126	18050	17754	18190
9	18194	17810	18076	18146	17382	17972	18012	18026	17892	17994
10	18112	18172	17986	17984	17668	17972	17594	18354	17946	18346
11	18030	18214	18456	18114	18076	17692	18290	18354	17882	18068
12	18144	17990	17804	17816	18092	17828	17966	18054	17750	18224
13	18122	18434	17974	18152	17862	17848	18116	17932	17938	17960
14	17966	18236	18220	18104	17954	18216	18180	18196	18412	18180
15	18372	18182	18130	18192	18214	18200	17932	18084	18188	18092
16	17748	18146	18514	17972	18136	17864	17898	17730	18162	18002
17	17966	17812	18052	18066	18162	18490	17976	17532	17936	17830
18	18372	18074	17934	18346	18416	17808	18032	18330	18204	18050
19	18170	17752	18246	18104	18198	18006	17824	17982	18140	18050
20	18154	18260	18200	18448	18246	18480	17986	17822	18290	18130
21	17972	18454	18030	17816	18164	18212	18320	18090	17990	18030
22	18076	18104	18128	18262	18292	17960	18268	18046	17642	18300
23	18732	18392	18546	17656	18300	18454	18246	18220	18294	18118
24	17728	18412	17882	18428	18276	18208	18402	18154	18272	18038
25	17806	18340	17954	18002	18132	18354	17920	18044	18080	17788
26	17838	18040	18318	17768	18406	18226	17778	18124	18202	18106
27	17864	17652	17658	17664	18366	18434	18058	18306	17952	17820

Tabla 29. Respuestas obtenidas por los veintisiete tratamientos en diez corridas para la instancia sko42

	Corrida									
Trat	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	1278456	1283612	1284374	1305444	1277318	1293052	1306852	1291470	1281852	1299940
2	1307626	1310920	1293218	1279058	1295980	1304670	1314530	1285152	1286346	1300990
3	1294570	1297524	1293810	1304376	1292460	1268682	1309100	1279034	1293300	1277790
4	1309116	1283456	1296886	1285518	1281910	1318906	1305840	1315882	1280300	1286472
5	1256986	1307572	1292310	1318018	1306074	1306074	1276444	1319708	1293376	1296560
6	1300990	1307788	1292376	1276304	1253080	1297002	1269684	1303340	1303888	1307398
7	1304134	1295782	1272950	1295232	1321520	1291848	1273160	1306158	1277300	1279970
8	1286636	1290752	1268966	1269684	1298320	1285076	1300938	1290200	1322168	1293784
9	1280218	1285604	1288580	1289694	1293424	1294270	1279296	1291240	1290804	1293222
10	1286466	1300594	1295974	1298002	1311326	1285192	1298992	1308634	1285596	1292124
11	1295842	1295204	1300222	1276658	1281762	1296512	1311026	1285250	1306054	1306886
12	1301888	1307202	1315112	1311928	1311460	1287102	1302792	1293168	1289136	1305000
13	1291882	1305968	1254170	1309574	1284776	1299594	1296208	1306062	1299568	1318660
14	1323078	1292142	1299452	1287298	1304120	1290972	1300876	1266288	1317042	1310454
15	1289640	1310818	1284910	1306368	1288394	1286234	1312200	1314376	1312256	1268320
16	1289996	1297214	1292418	1288384	1301428	1286416	1282294	1263302	1270730	1316080
17	1276728	1307174	1293348	1309474	1282096	1296750	1281126	1301608	1278318	1300376
18	1302044	1310298	1284712	1278946	1291006	1307486	1308286	1289992	1298738	1309060
19	1287032	1278030	1290216	1293740	1297340	1281428	1317500	1290772	1301400	1297744
20	1303604	1294440	1284524	1280248	1292110	1298374	1297400	1276628	1302384	1299230
21	1302588	1303518	1308474	1293158	1299232	1274124	1274370	1310062	1301438	1309462
22	1259516	1299058	1312532	1294820	1299582	1316960	1306880	1305020	1306514	1260484
23	1326204	1301626	1309478	1304210	1283078	1289208	1298130	1290454	1302810	1285290
24	1308504	1273442	1278844	1301328	1291420	1282420	1287570	1303320	1308974	1298716
25	1271438	1302332	1302404	1294264	1292254	1274470	1283982	1259654	1298682	1277016
26	1293140	1306798	1308492	1315780	1293142	1283678	1291376	1278876	1286132	1313370
27	1294626	1315688	1301156	1296754	1292464	1309552	1264520	1301434	1295452	1312758

Tabla 30. Respuestas obtenidas por los veintisiete tratamientos en diez corridas para la instancia tai25a

Los resultados de las tablas ANDEVA que se obtuvieron con el Proc GLM se sintetizan en la tabla 27. Para cada una de las instancias seleccionadas se identifican los factores e interacciones dobles que fueron significativos. La interacción triple 1*2*3 (**evapor*divers*explor**) fue únicamente significativa para la instancia bur26e y por eso no se menciona en la tabla. De acuerdo con la columna dos el factor más significativo en la respuesta entregada por HAS-QAP^R, es el de diversificación (2), con presencia en todas las instancias, excepto en la tai20b. Las interacciones dobles consideradas significativas, fueron 2*3 (**divers*explor**) y 1*3 (**evapor*divers**). La columna cuatro ordena de menor a mayor los promedios de los desempeños obtenidos con cada uno de los niveles de los factores. La última columna da la secuencia de las

mejores combinaciones de los niveles para las interacciones dobles. Los niveles del factor diversificación con las mejores respuestas en la mayoría de las instancias fueron el 3 y el 1, con promedios similares. Así mismo, las combinaciones con las mejores respuestas para la interacción 2*3 fueron 3-3, 3-1, 3-2 y 1-3 y para la 1*3, 3-3 y 1-3. Lo anterior parece indicar que los mejores desempeños del algoritmo se logran con niveles opuestos del factor diversificación, es decir, con el máximo ($>> n$), cuando la diversificación no se implementa, y con el mínimo ($n/2$), cuando la probabilidad de que se implemente es la mayor. Por otra parte, las respuestas que se obtienen con estos niveles del factor diversificación en la interacción doble 2*3 son insensibles a los niveles que toma el factor exploración, con buenas respuestas para cualquiera de las combinaciones. Sin embargo, las mejores combinaciones de la interacción 1*3 indican que existe una notable diferencia en las respuestas obtenidas, cuando éste factor (exploración) tiende a 1 (sólo intensificación-regla de explotación). El factor evaporación es el menos significativo, y esta última interacción (interacción 1*3) refleja que el desempeño alcanzado para una instancia determinada es independiente del nivel que este tome, con buenas respuestas para las combinaciones formadas manteniendo fijo el nivel de exploración (específicamente los niveles 2 (0,9) y 3 (1) -los mejores-) con cualquiera de los niveles del factor exploración.

Instancia	Factor	Interacción doble	Mejor Nivel Factor	Mejor Combinación Interacción
Bur26a	1 2	2*3	3,2,1 3,1,2	3-3,3-1
Chr25	3 2	1*3 2*3	2,1,3 3,2,1	3-3,1-3, 2-3,1-1 3-2,3-1,2-2
Kr30	2	1*3	3,1,2	2-1,3-3,1-3
tai20b	-	-	-	-
Nug20	2	2*3	3,1,2	3-1,3-3, 1-3, 3-2
sko42	3 2	-	3,1,2 1,3,2	-
tai25a	2	-	3,1,2	-

Tabla 31. Resumen de los resultados de las tablas ANDEVA para las instancias seleccionadas

El análisis de duncan (arrojado por el Proc GLM) dio los promedios de las respuestas de los veintisiete tratamientos para cada una de las instancias seleccionadas. Para cada instancia se identificaron tres grupos de tratamientos: bajo, medio y alto, de acuerdo al desempeño alcanzado por el tratamiento, dado en función de su promedio. Los valores en negrilla están asociados al peor (promedio más alto) y mejor (promedio más bajo) tratamiento respectivamente.

El análisis de duncan refleja que el efecto conjunto de los tres mecanismos sobre el desempeño alcanzado por HAS-QAP no es significativo. Los grupos definidos para la clasificación de los tratamientos no están integrados por combinaciones de niveles específicos sino por varios de ellos. Es así como en un mismo grupo pueden encontrarse tratamientos formados por la combinación de los niveles más bajos hasta aquellos formados por la combinación de los más altos. De ese modo, el análisis valida la interpretación dada para la interacción triple **evapor*divers*explor** no significativa encontrada para la mayoría de las instancias seleccionadas, y es el que no existen diferencias significativas entre los promedios entregados por los tratamientos. De lo anterior se deduce que el desempeño obtenido por el algoritmo HAS-QAP^R para una instancia específica es insensible a los niveles que tomen los mecanismos de evaporación, diversificación y exploración simultáneamente, consiguiendo para cualquier combinación de ellos, es decir, para cualquier tratamiento, respuestas similares, como se alcanza a observar en las gráficas con una serie casi suavizada representando el comportamiento de los promedios de los tratamientos.

Aún así, el análisis de los resultados entregados por las tablas ANDEVA encuentra que dentro de todos los tratamientos el empleado para implementar y poner a prueba a HAS-QAP^R puede ser considerado uno de los mejores.

10.1. Diseño factorial instancias categoría 1

- **Instancia bur26a**

La tabla ANDEVA que arrojó el Proc GLM para esta instancia fue la siguiente.

The GLM Procedure
Dependent Variable: respuesta

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	35	13548947270	387112779	1.67	0.0144
Error	234	54233958604	231769054		
Corrected Total	269	67782905873			

R-Square	Coeff Var	Root MSE	respuesta Mean
0.199887	0.278084	15223.96	5474588

Source	DF	Type I SS	Mean Square	F Value	Pr > F
corrida	9	2367499080	263055453	1.13	0.3387
evaporacion	2	1414341022	707170511	3.05	0.0492
diversificacion	2	2988130759	1494065380	6.45	0.0019
exploracion	2	550273484	275136742	1.19	0.3069
evaporaci*diversific	4	400705409	100176352	0.43	0.7853
evaporaci*exploracio	4	304589402	76147350	0.33	0.8586
diversifi*exploracio	4	2296261386	574065347	2.48	0.0450
evapor*divers*explor	8	3227146727	403393341	1.74	0.0900

Las conclusiones que se sacaron a partir de la tabla son.

1. El factor bloque no es significativo y por ende no hay diferencias significativas entre los promedios de las corridas. Esto significa que no hay diferencias significativas en las respuestas que obtuvo cada tratamiento entre corridas.

2. El Valor de F para **evaporacion** y **diversificacion** es significativo y en consecuencia los promedios de los dos factores son diferentes. Esto implica que la calidad de la respuesta esta influenciada por ambos. El valor de F para **exploracion** no es significativo y por ende no hay diferencias significativas entre los promedios poblacionales de sus tres niveles.

3. Las interacciones **evaporaci*exploracio** y **evaporaci*diversificac** al no ser significativas, aseguran para los factores exploración y diversificación que sus tres niveles entre ellos mantengan las mismas diferencias relativas sin importar el nivel

que tome el factor evaporación. Esto significa que el desempeño de los mecanismos de diversificación y exploración es independiente del de evaporación.

4. La interacción **exploracio*diversificac** es significativa, esto indica que los factores exploración y diversificación son dependientes, y que el nivel del factor diversificación que tenga asociado el mejor promedio depende del nivel que tome el factor exploración.

5. La interacción triple **evapor*divers*explor** es significativa. Esto significa que hay diferencias significativas en los promedios de las respuestas entregadas por los veintisiete tratamientos.

La clasificación de los tratamientos de acuerdo al análisis de duncan esta dada en la tabla 32, de acuerdo con ella el peor tratamiento es el 23 y el mejor el 27 (resaltados en negrilla). La figura 30 muestra el comportamiento de los promedios en el orden en que están secuenciados en la tabla 32 (de mayor a menor promedio). De esa forma, el primer punto en la gráfica corresponde al promedio del peor tratamiento y el último al promedio del mejor.

Grupo	Trat	Promedio
Bajo	**23**	**5487703,5**
	4	5486152,3
	12	5483948
	14	5482523,2
	3	5482276,1
	17	5480033
Medio	8	5479333,5
	6	5478364,5
	13	5477938,7
	5	5477345,5
	19	5476927,1
	2	5475401,4
	24	5474914,4
	7	5474351,4
	10	5473907,5
	15	5472522,3
	1	5471408,3
	11	5471407,8

Alto	22	5469880
	21	5469160,9
	18	5468991
	20	5468064,7
	26	5467571
	9	5467500,1
	16	5467229,7
	25	5465930,3
	27	**5463094**

Tabla 32. Agrupación de los tratamientos de acuerdo a los promedios arrojados por el análisis de duncan para la instancia bur26e

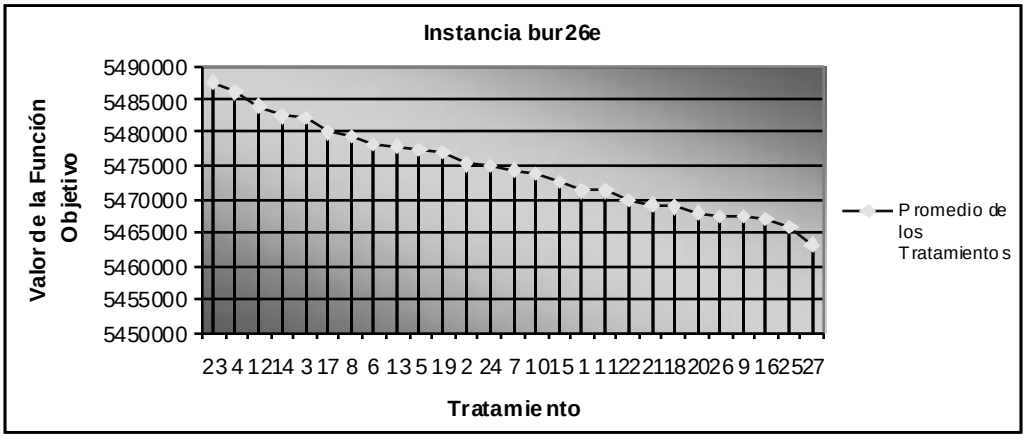


Figura 30. Gráfica del comportamiento de los promedios de los tratamientos para la instancia bur26e

- **Instancia chr25**

La tabla ANDEVA que arrojó el Proc GLM para esta instancia fue la siguiente.

The GLM Procedure
Dependent Variable: respuesta

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	35	49441880.4	1412625.2	1.96	0.0018
Error	234	168610516.6	720557.8		
Corrected Total	269	218052397.1			

R-Square	Coeff Var	Root MSE	respuesta Mean
0.226743	9.534994	848.8567	8902.541

Source	DF	Type I SS	Mean Square	F Value	Pr > F
corrida	9	6386282.98	709587.00	0.98	0.4533
evaporacion	2	770217.27	385108.64	0.53	0.5867
diversificacion	2	8136601.10	4068300.55	5.65	0.0040

exploracion	2	4356490.87	2178245.44	3.02	0.0506
evaporaci*diversific	4	2928848.41	732212.10	1.02	0.3997
evaporaci*exploracio	4	6680161.84	1670040.46	2.32	0.0579
diversifi*exploracio	4	14272355.88	3568088.97	4.95	0.0008
evapor*divers*explor	8	5910922.07	738865.26	1.03	0.4175

Las conclusiones que se sacaron a partir de la tabla son.

1. El factor bloque no es significativo y por ende no hay diferencias significativas entre los promedios de las corridas. Esto significa que no hay diferencias significativas en las respuestas que obtuvo cada tratamiento entre corridas.

2. El Valor de F para **exploracion** y **diversificacion** es significativo y en consecuencia los promedios de los dos factores son diferentes. Esto implica que la calidad de la respuesta esta influenciada por ambos. El valor de F para **evaporacion** no es significativo y por ende no hay diferencias significativas entre los promedios de sus tres niveles.

3. Las interacción **evaporaci*diversificac** al no ser significativa, asegura para el factor diversificación que sus tres niveles entre ellos mantengan las mismas diferencias relativas sin importar el nivel que tome el factor evaporación. Esto significa que el desempeño del mecanismo de diversificación no depende del mecanismo de evaporación.

4. Las interacciones **evaporaci*exploracio** y **diversifi*exploracio** son significativas. Por tanto, el nivel del factor exploración que tenga asociado el mejor promedio depende del nivel que tomen los factores evaporación y diversificación, y el nivel del factor diversificación correspondiente al mejor promedio depende del nivel que tome el factor exploración.

5. La interacción triple **evapor*divers*explor** no es significativa. Esto significa que no hay diferencias significativas entre los promedios entregados por los veintisiete tratamientos.

La clasificación de los tratamientos de acuerdo al análisis de duncan esta dada en la tabla 33. El peor y el mejor tratamiento para esta instancia son el 4 y el 26 respectivamente. La gráfica con el comportamiento de los promedios se observa en la figura 31.

Grupo	Trat	Promedio
Bajo	**4**	**9651,4**
	22	9440
	24	9362
	13	9350,8
	20	9307
	15	9269,4
	27	9222
	10	9067,8
	12	9062
	18	9054,8
	21	9053,4
	2	9045,6
	11	9032
	19	8983,2
	1	8909,6
Medio	3	8878
	5	8874,4
	16	8853,8
	17	8757,4
	9	8733,2
	8	8663
	7	8531,4
Alto	23	8451
	6	8375
	14	8338,2
	25	8061
	26	**8039**

Tabla 33. Agrupación de los tratamientos de acuerdo a los promedios arrojados por el análisis de duncan para la instancia chr25

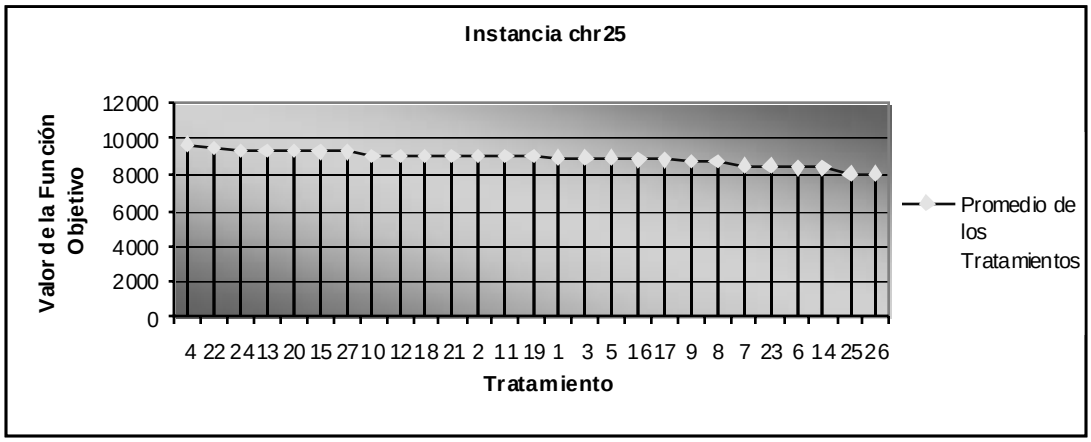


Figura 31. Gráfica del comportamiento de los promedios de los tratamientos para la instancia chr25

- **Instancia kr30a**

La tabla ANDEVA que arrojó el Proc GLM para esta instancia fue la siguiente.

The GLM Procedure
Dependent Variable: respuesta

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	35	347735020	9935286	1.06	0.3836
Error	234	2190892995	9362791		
Corrected Total	269	2538628015			

R-Square	Coeff Var	Root MSE	respuesta Mean
0.136978	2.904643	3059.868	105344.0

Source	DF	Type I SS	Mean Square	F Value	Pr > F
corrida	9	59265587.1	6585065.2	0.70	0.7056
evaporacion	2	4050358.0	2025179.0	0.22	0.8057
diversificacion	2	42907531.9	21453765.9	2.29	0.1034
exploracion	2	1097445.7	548722.9	0.06	0.9431
evaporaci*diversific	4	120006537.6	30001634.4	3.20	0.0138
evaporaci*exploracio	4	44580107.0	11145026.8	1.19	0.3157
diversifi*exploracio	4	17844321.5	4461080.4	0.48	0.7530
evapor*divers*explor	8	57983130.7	7247891.3	0.77	0.6259

Las conclusiones que se sacaron a partir de la tabla son.

1. El factor bloque no es significativo y por ende no hay diferencias significativas entre los promedios de las corridas. Esto significa que no hay diferencias significativas en las respuestas que obtuvo cada tratamiento entre corridas.

2. Sólo el valor de F para **diversificacion** es significativo y en consecuencia los promedios de este factor son diferentes. Esto implica que la calidad de la respuesta esta influenciada por este factor. El valor de F para el resto de factores no es significativo y por ende no hay diferencias significativas entre los promedios de sus tres niveles.

3. La única interacción doble significativa es la de **evaporaci*diversific**, esto indica que el nivel del factor diversificación correspondiente al mejor promedio depende del nivel que tome el factor evaporación.

4. La interacción triple **evapor*divers*explor** no es significativa.

La tabla 34 da la clasificación de los tratamientos de acuerdo al análisis de duncan. El peor tratamiento es el 18 y el mejor el 10. La figura 32 corresponde a la gráfica del comportamiento de los promedios de los tratamientos.

Grupo	Trat	Promedio
Bajo	**18**	**107362**
	13	106788
	6	106437
	4	106346
	15	106332
	1	106023
	20	106003
	17	105883
	23	105813
Medio	7	105696
	21	105655
	2	105635
	5	105621
	19	105472
	22	105375
	14	105276
	8	105232
	3	105201
	24	105179

	16	105068
	25	105039
Alto	11	104656
	26	104442
	12	104350
	27	103659
	9	103111
	10	**102628**

Tabla 34. Agrupación de los tratamientos de acuerdo a los promedios arrojados por el análisis de duncan para la instancia kr30a

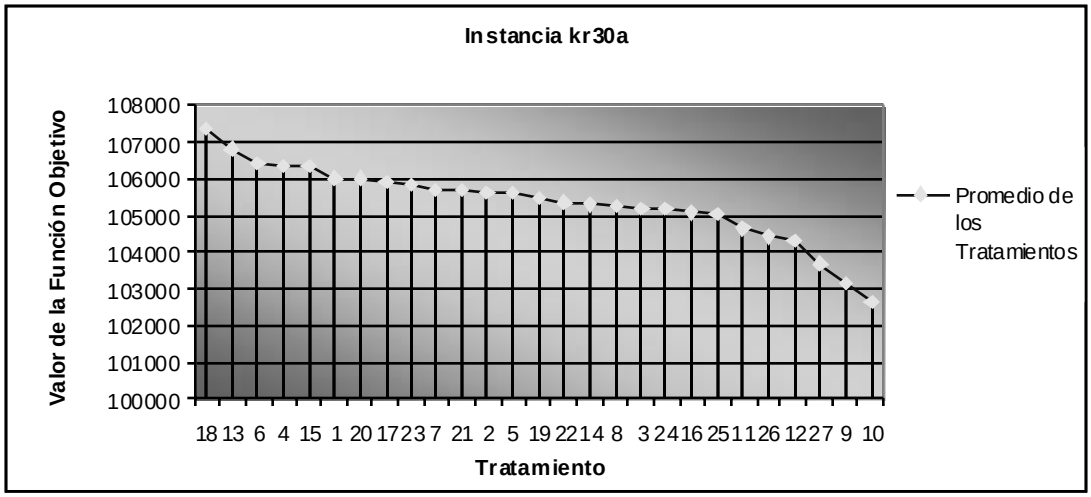


Figura 32. Gráfica del comportamiento de los promedios de los tratamientos para la instancia kr30a

- **Instancia tai20b**

La tabla ANDEVA que arrojó el Proc GLM para esta instancia fue la siguiente.

```
The GLM Procedure
Dependent Variable: respuesta

Source                      DF    Sum of Squares      Mean Square     F Value     Pr > F
Model                       35     1.9987458E17     5.7107023E15        0.97     0.5227
Error                      234     1.3784611E18     5.8908593E15
Corrected Total            269     1.5783356E18

R-Square     Coeff Var      Root MSE     respuesta Mean
0.126636      55.87331      76751933          137367802

Source                      DF       Type I SS     Mean Square    F Value     Pr > F
```

corrida	9	4.9934022E16	5.5482247E15	0.94	0.4895
evaporacion	2	8.5145061E15	4.257253E15	0.72	0.4865
diversificacion	2	8.0907938E15	4.0453969E15	0.69	0.5042
exploracion	2	8.489492E15	4.244746E15	0.72	0.4876
evaporaci*diversific	4	2.7646827E16	6.9117067E15	1.17	0.3233
evaporaci*exploracio	4	2.9883372E16	7.470843E15	1.27	0.2832
diversifi*exploracio	4	2.6261391E16	6.5653477E15	1.11	0.3503
evapor*divers*explor	8	4.1054176E16	5.131772E15	0.87	0.5415

Las conclusiones que se sacaron a partir de la tabla son.

1. El factor bloque no es significativo y por ende no hay diferencias significativas entre los promedios de las corridas. Esto significa que no hay diferencias significativas en las respuestas que obtuvo cada tratamiento entre corridas.

2. El valor de F para todos los factores es no significativo y en consecuencia no hay diferencias significativas entre los promedios de sus tres niveles. Esto significa que la calidad de la respuesta es independiente del nivel que tomen los factores.

3. Ninguna de las interacciones dobles es significativa. Esto significa que el desempeño de los mecanismos es independiente el uno del otro.

4. La interacción triple **evapor*divers*explor** no es significativa.

La tabla 35 da la clasificación de los tratamientos de acuerdo al análisis de duncan. El peor tratamiento es el 26 y el mejor el 10. La figura 33 corresponde a la gráfica del comportamiento de los promedios de los tratamientos

Grupo	Trat	Promedio
Bajo	**26**	**246487279**
	6	182379300
	20	135789131
	2	135110141
	25	133440846
	3	133368348
	15	133283702

	8	133028391
Medio	18	132623135
	13	132356584
	16	132176283
	22	131920781
	11	131186005
	7	130985780
	9	130956454
	27	130952519
	4	130931590
	23	130695668
	5	130626864
Alto	24	129888424
	19	129685740
	21	129465837
	14	129116011
	1	128689357
	12	128472325
	17	128392117
	10	**126922055**

Tabla 35. Agrupación de los tratamientos de acuerdo a los promedios arrojados por el análisis de duncan para la instancia tai20b

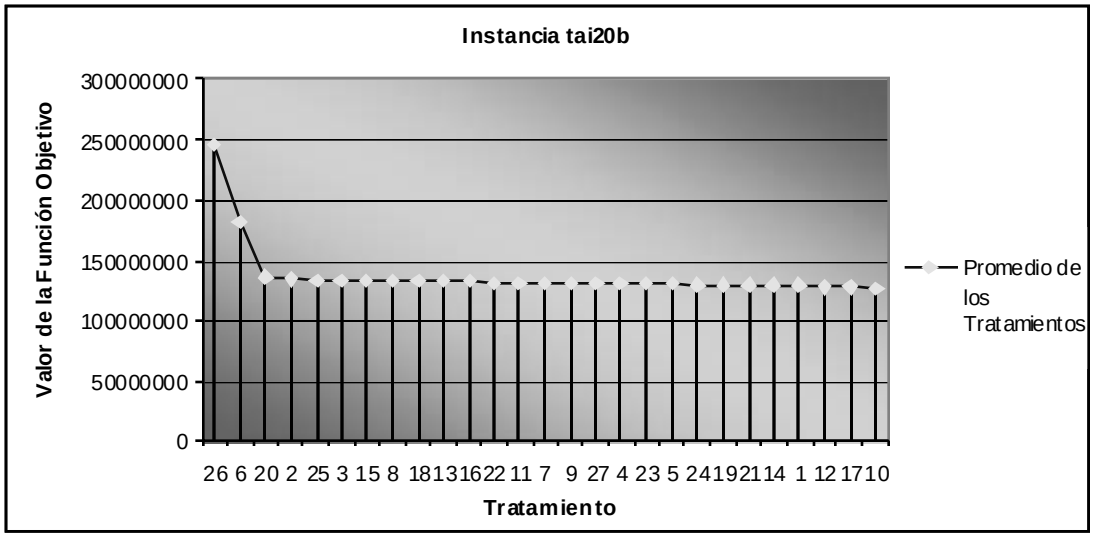


Figura 33. Gráfica del comportamiento de los promedios de los tratamientos para la instancia tai20b

10.2. Diseño factorial instancias categoría 2

- **Instancia nug20**

La tabla ANDEVA que arrojó el Proc GLM para esta instancia fue la siguiente.

The GLM Procedure
Dependent Variable: respuesta

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	35	73385.6741	2096.7335	1.01	0.4537
Error	234	484015.5259	2068.4424		
Corrected Total	269	557401.2000			

R-Square	Coeff Var	Root MSE	respuesta Mean
0.131657	1.649305	45.48013	2757.533

Source	DF	Type I SS	Mean Square	F Value	Pr > F
corrida	9	20515.27407	2279.47490	1.10	0.3621
evaporacion	2	1830.75556	915.37778	0.44	0.6429
diversificacion	2	18629.95556	9314.97778	4.50	0.0120
exploracion	2	289.86667	144.93333	0.07	0.9323
evaporaci*diversific	4	2847.55556	711.88889	0.34	0.8479
evaporaci*exploracio	4	3677.51111	919.37778	0.44	0.7764
diversifi*exploracio	4	13726.31111	3431.57778	1.66	0.1603
evapor*divers*explor	8	11868.44444	1483.55556	0.72	0.6762

Las conclusiones que se sacaron a partir de la tabla son.

1. El factor bloque no es significativo y por ende no hay diferencias significativas entre los promedios de las corridas. Esto significa que no hay diferencias significativas en las respuestas que obtuvo cada tratamiento entre corridas.

2. Sólo el valor de F para **diversificacion** es significativo y en consecuencia los promedios de sus tres niveles son diferentes. Esto implica que la calidad de la respuesta esta influenciada por este factor. El valor de F para el resto de factores no es significativo y por ende no hay diferencias significativas entre sus promedios.

3. La única interacción doble significativa es la **diversifi*exploracio**, esto indica que el nivel del factor diversificación correspondiente al mejor promedio depende del nivel que tome el factor exploración.

4. La interacción triple **evapor*divers*explor** no es significativa.

La clasificación de los tratamientos de acuerdo al análisis de duncan se da en la tabla 36. El peor y el mejor tratamiento para esta instancia son el 4 y el 16 respectivamente. La gráfica con el comportamiento de los promedios se observa en la figura 34.

Grupo	Trat	Promedio
Bajo	**4**	**2785,2**
	6	2780,8
	24	2775,6
	10	2774,8
	13	2773
	14	2767,2
	11	2763,4
	19	2763,2
	15	2762,2
	17	2762,2
	1	2762
Medio	20	2760,2
	8	2758,2
	22	2756,4
	2	2755
	5	2754,4
	7	2753,6
	3	2753
	18	2752,6
	12	2752,2
	23	2748,2
	26	2747,6
	25	2747
Alto	21	2744
	27	2744
	9	2741
	16	**2714,4**

Tabla 36. Agrupación de los tratamientos de acuerdo a los promedios arrojados por el análisis de duncan para la instancia nug20

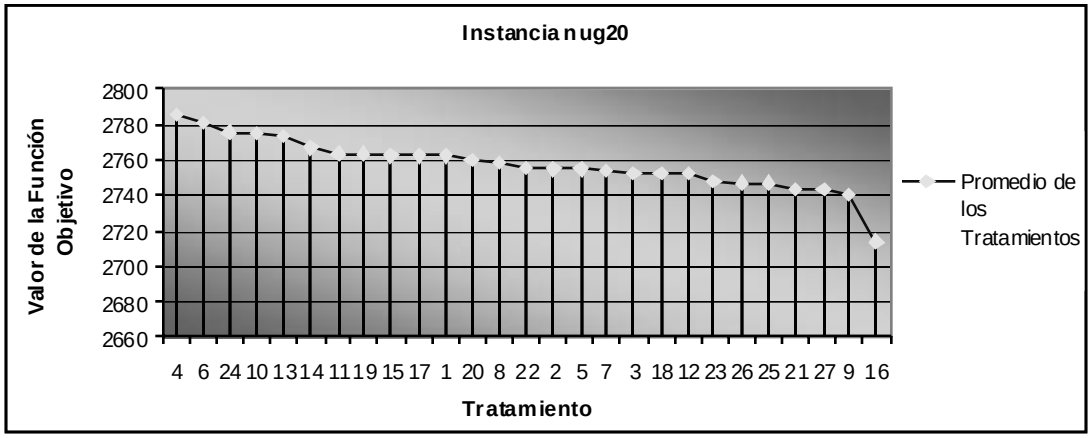


Figura 34. Gráfica del comportamiento de los promedios de los tratamientos para la instancia nug20

- **Instancia sko42**

La tabla ANDEVA que arrojó el Proc GLM para esta instancia fue la siguiente.

The GLM Procedure
Dependent Variable: respuesta

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	35	2212831.97	63223.77	1.46	0.0542
Error	234	10131871.97	43298.60		
Corrected Total	269	12344703.94			

R-Square	Coeff Var	Root MSE	respuesta Mean
0.179254	1.149884	208.0832	18096.01

Source	DF	Type I SS	Mean Square	F Value	Pr > F
observacion	9	249720.8296	27746.7588	0.64	0.7615
corrida	2	111070.7852	55535.3926	1.28	0.2793
diversificacion	2	779598.2519	389799.1259	9.00	0.0002
exploracion	2	210999.3185	105499.6593	2.44	0.0897
evaporaci*diversific	4	165445.5704	41361.3926	0.96	0.4329
evaporaci*exploracio	4	164611.4370	41152.8593	0.95	0.4356
diversifi*exploracio	4	152097.0370	38024.2593	0.88	0.4777
evapor*divers*explor	8	379288.7407	47411.0926	1.09	0.3674

Las conclusiones que se sacaron a partir de la tabla son.

1. El factor bloque no es significativo y por ende no hay diferencias significativas entre los promedios de las corridas. Esto significa que no hay diferencias significativas en las respuestas que obtuvo cada tratamiento entre corridas.

2. El valor de F para **exploracion** y **diversificacion** es significativo y en consecuencia los promedios de los factores son diferentes. Esto implica que la calidad de la respuesta esta influenciada por ambos factores. El valor de F para **evaporacion** no es significativo y por ende no hay diferencias significativas entre los promedios de sus tres niveles.

3. Ninguna de las interacciones dobles es significativa. Esto significa que el desempeño de los mecanismos es independiente de los niveles que tomen.

4. La interacción triple **evapor*divers*explor** no es significativa.

La tabla 37 da la clasificación de los tratamientos de acuerdo al análisis de duncan. El peor y el mejor tratamiento para esta instancia son el 23 y el 9 respectivamente. La gráfica con el comportamiento de los promedios se observa en la figura 35.

Grupo	Trat	Promedio
Bajo	**23**	**18295,8**
	20	18201,6
	3	18187,8
	6	18184,4
	24	18180
	4	18174
	5	18172,6
	14	18166,4
	15	18158,6
	18	18156,6
	2	18130,2
Medio	11	18117,6
	21	18107,8
	22	18107,8
	26	18080
	7	18061,6
	1	18056,2
	19	18047,2
	25	18042
	13	18033,8
	8	18022,4
	16	18017,2
	10	18013,4
Alto	17	17982,2
	27	17977,4
	12	17966,8
	9	**17950,4**

Tabla 37. Agrupación de los tratamientos de acuerdo a los promedios arrojados por el análisis de duncan para la instancia sko42

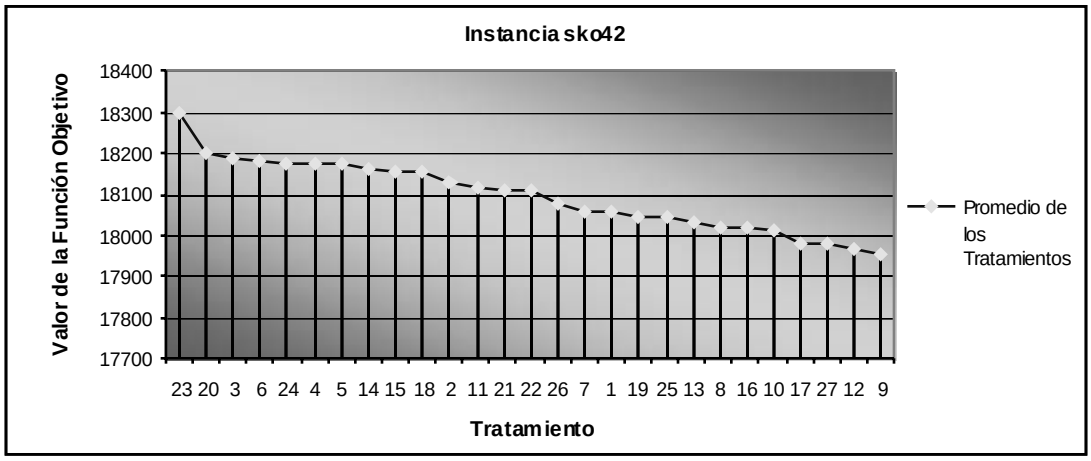


Figura 35. Gráfica del comportamiento de los promedios de los tratamientos para la instancia sko42

- **Instancia tai25a**

La tabla ANDEVA que arrojó el Proc GLM para esta instancia fue la siguiente.

The GLM Procedure
Dependent Variable: respuesta

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	35	5043710697	144106020	0.73	0.8644
Error	234	46009153020	196620312		
Corrected Total	269	51052863717			

R-Square	Coeff Var	Root MSE	respuesta Mean
0.098794	1.083066	14022.14	1294670

Source	DF	Type I SS	Mean Square	F Value	Pr > F
observacion	9	959002373.7	106555819.3	0.54	0.8430
corrida	2	570714082.8	285357041.4	1.45	0.2364
diversificacion	2	726468526.0	363234263.0	1.85	0.1599
exploracion	2	461966438.0	230983219.0	1.17	0.3107
evaporaci*diversific	4	137522969.4	34380742.4	0.17	0.9512
evaporaci*exploracio	4	790084622.5	197521155.6	1.00	0.4058
diversifi*exploracio	4	693808999.6	173452249.9	0.88	0.4753
evapor*divers*explor	8	704142685.1	88017835.6	0.45	0.8913

Las conclusiones que se sacaron a partir de la tabla son.

1. El factor bloque no es significativo y por ende no hay diferencias significativas entre los promedios de las corridas. Esto significa que no hay diferencias significativas en las respuestas que obtuvo cada tratamiento entre corridas.

2. Sólo el valor de F para **diversificacion** es significativo y en consecuencia los promedios de este factor son diferentes. Esto implica que la calidad de la respuesta esta influenciada por este factor. El valor de F para el resto de factores no es significativo y por ende no hay diferencias significativas entre los promedios de sus tres niveles.

3. Ninguna de las interacciones dobles es significativa. Esto significa que el desempeño de los mecanismos es independiente de los niveles que tomen.

4. La interacción triple **evapor*divers*explor** no es significativa.

La tabla 38 da la clasificación de los tratamientos de acuerdo al análisis de duncan. El peor y el mejor tratamiento para esta instancia son el 12 y el 25 respectivamente.

La gráfica con el comportamiento de los promedios se observa en la figura 36.

Grupo	Trat	Promedio
Bajo	**12**	**1302478,8**
	14	1299172,2
	23	1299048,8
	27	1298440,4
	18	1298056,8
	2	1297849
	21	1297642,6
	15	1297351,6
	5	1297312,2
	26	1297078,4
	13	1296646,2
	4	1296428,6
	10	1296290
	22	1296136,6
Medio	11	1295541,6
	19	1293520,2
	24	1293453,8
	20	1292894,2
	17	1292699,8
	7	1291805
	6	1291185
	3	1291064,6
Alto	8	1290652,4
	1	1290237
	16	1288826,2
	9	1288635,2
	25	**1285649,6**

Tabla 38. Agrupación de los tratamientos de acuerdo a los promedios arrojados por el análisis de duncan para la instancia tai25a

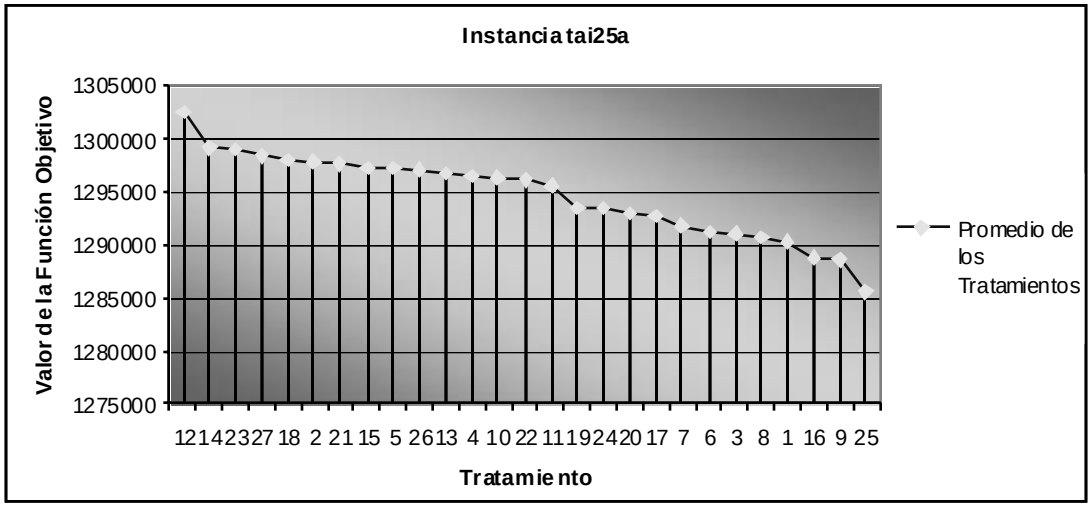


Figura 36. Gráfica del comportamiento de los promedios de los tratamientos para la instancia tai25a

Capítulo XI

ANÁLISIS DE VARIANZA

En esta sección se aplica un procedimiento de regresión lineal con el fin de analizar la relación entre la respuesta (variable dependiente) entregada por HAS-QAP^R y los mecanismos seleccionados en el diseño factorial (variables independientes), evaporación, diversificación y exploración. El procedimiento utilizado fue el Proc REG del programa SAS por su afinidad con el Proc GLM. El análisis de regresión tiene como propósito confirmar los resultados del diseño factorial para cada una de las instancias, como determinar que tanto esta explicada la variabilidad de las respuestas por los tres mecanismos simultáneamente, es decir, si la relación entre la respuesta y los tres mecanismos puede ser considerada lineal. Otras de las informaciones que genera este procedimiento son los estimados de los mecanismos, y su efecto independiente en la respuesta. Los resultados entregados por el Proc REG para cada una de las instancias se muestran a continuación.

- **Instancia bur26e**

La tabla ANOVA que arrojó el Proc REG para esta instancia fue la siguiente.

The REG Procedure
Model: regresion
Dependent Variable: respuesta

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	3	2154659332	718219777	2.91	0.0350
Error	266	65628246541	246722731		
Corrected Total	269	67782905873			

Root MSE	15707	**R-Square**	0.0318
Dependent Mean	5474588	**Adj R-Sq**	0.0209
Coeff Var	0.28691		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	5484622	4166.76973	1316.28	<.0001
evaporacion	1	-2715.93333	1170.76122	-2.32	0.0211
diversificacion	1	-2137.08333	1170.76122	-1.83	0.0691
exploracion	1	-164.04444	1170.76122	-0.14	0.8887

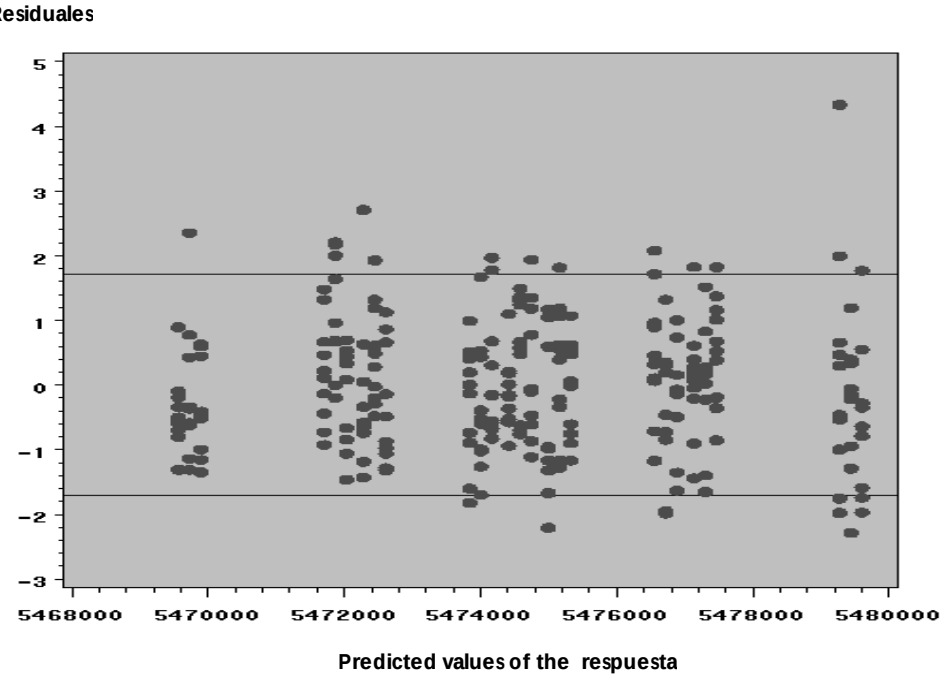


Figura 37. Gráfica de los residuales contra valores estimados de la respuesta para la instancia bur26e

- **Instancia chr25**

La tabla ANOVA que arrojó el Proc REG para esta instancia fue la siguiente.

The REG Procedure
Model: regresion
Dependent Variable: respuesta

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	3	6216655	2072218	2.75	0.0659
Error	23	17332286	753578		
Corrected Total	26	23548941			

Root MSE	868.08851	**R-Square**	0.2640
Dependent Mean	8980.51852	**Adj R-Sq**	0.1680
Coeff Var	9.66635		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	8943.18519	728.21383	12.28	<.0001
evaporacion	1	86.22222	204.61042	0.42	0.6774
diversificacion	1	375.88889	204.61042	1.84	0.0792
exploracion	1	-443.44444	204.61042	-2.17	0.0408

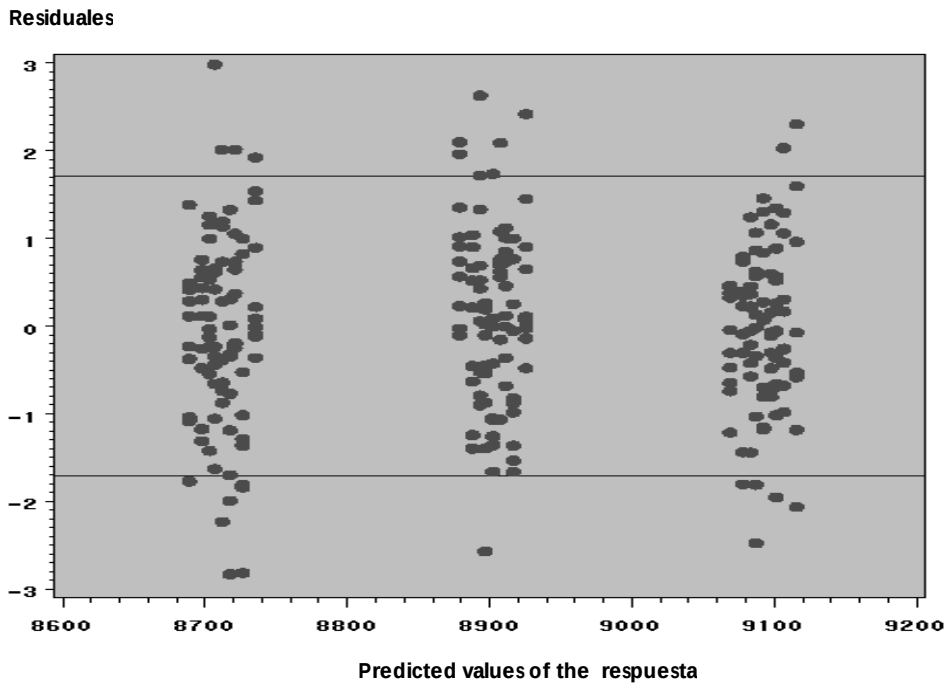


Figura 38. Gráfica de los residuales contra valores estimados de la respuesta para la instancia chr25

- **Instancia kr30a**

La tabla ANOVA que arrojó el Proc REG para esta instancia fue la siguiente.

The REG Procedure
Model: regresion
Dependent Variable: respuesta

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	3	18157036	6052345	0.70	0.5589
Error	23	197536381	8588538		
Corrected Total	26	215693417			

Root MSE	2930.62080	**R-Square**	0.0842
Dependent Mean	104984	**Adj R-Sq**	-0.0353
Coeff Var	2.79148		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	108438	2458.41132	44.11	<.0001
evaporacion	1	-478.88889	690.75395	-0.69	0.4951
diversificacion	1	-602.50000	690.75395	-0.87	0.3921
exploracion	1	-645.27778	690.75395	-0.93	0.3599

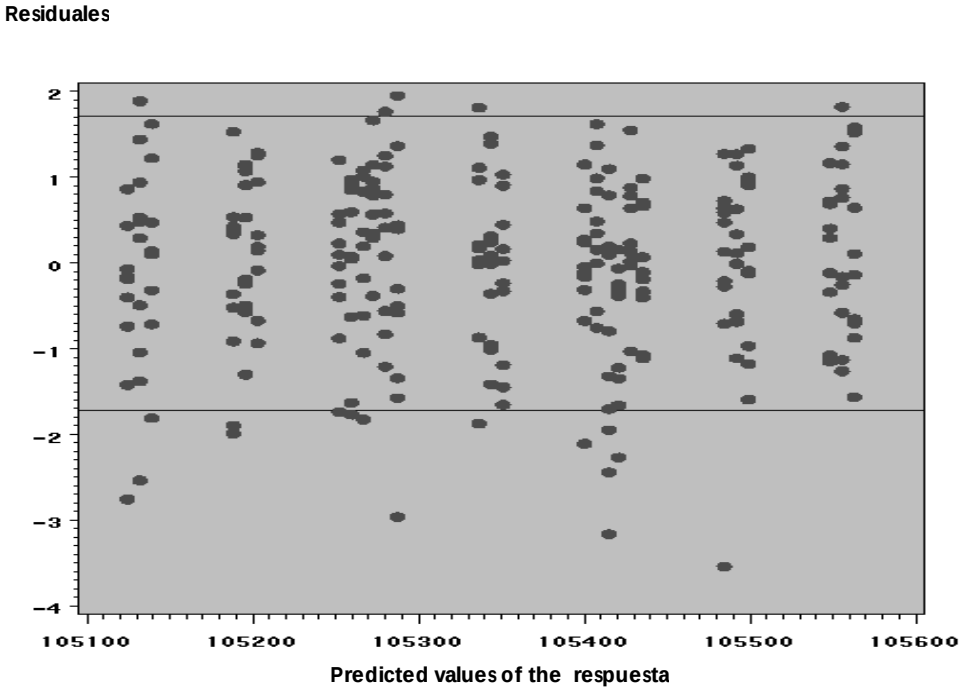


Figura 39. Gráfica de los residuales contra valores estimados de la respuesta para la instancia kr30a

- **Instancia tai20b**

La tabla ANOVA que arrojó el Proc REG para esta instancia fue la siguiente.

The REG Procedure
Model: regresion
Dependent Variable: respuesta

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	3	1.898045E14	6.326817E13	0.74	0.5396
Error	23	1.969248E15	8.561949E13		
Corrected Total	26	2.159053E15			

Root MSE	9253080	**R-Square**	0.0879
Dependent Mean	132633213	**Adj R-Sq**	-0.0311
Coeff Var	6.97644		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	128830529	7762136	16.60	<.0001
evaporacion	1	-1832540	2180972	-0.84	0.4094
diversificacion	1	2195237	2180972	1.01	0.3246
exploracion	1	1538645	2180972	0.71	0.4876

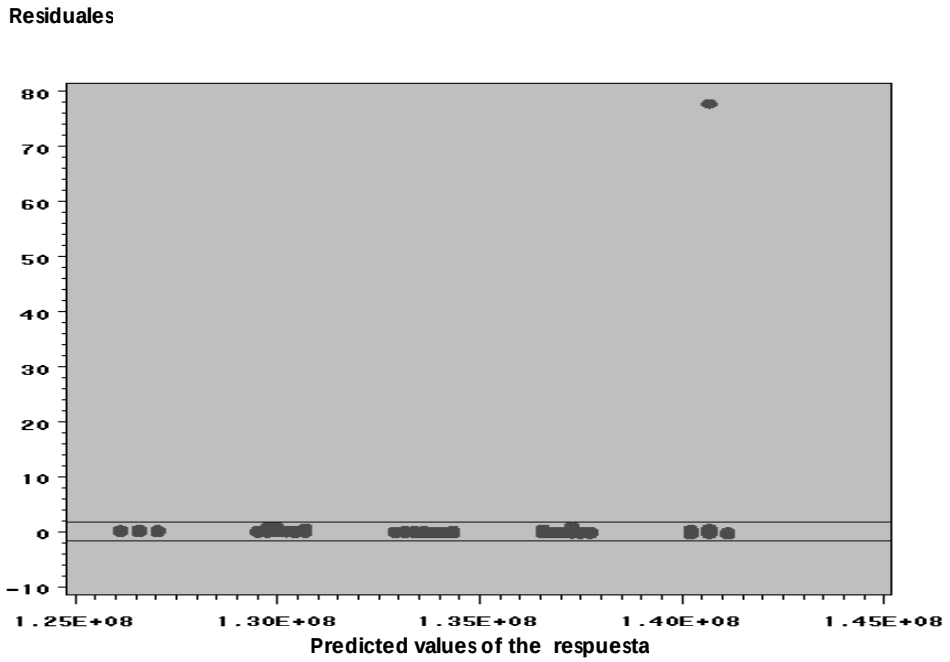


Figura 40. Gráfica de los residuales contra valores estimados de la respuesta para la instancia tai20b

- **Instancia nug20**

La tabla ANOVA que arrojó el Proc REG para esta instancia fue la siguiente.

The REG Procedure
Model: regresion
Dependent Variable: respuesta

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	3	8441.71111	2813.90370	1.36	0.2543
Error	266	548959	2063.75748		
Corrected Total	269	557401			

Root MSE	45.42860	**R-Square**	0.0151
Dependent Mean	2757.53333	**Adj R-Sq**	0.0040
Coeff Var	1.64744		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	2778.26667	12.05103	230.54	<.0001
evaporacion	1	-3.15556	3.38605	-0.93	0.3522
diversificacion	1	-5.94444	3.38605	-1.76	0.0803
exploracion	1	-1.26667	3.38605	-0.37	0.7086

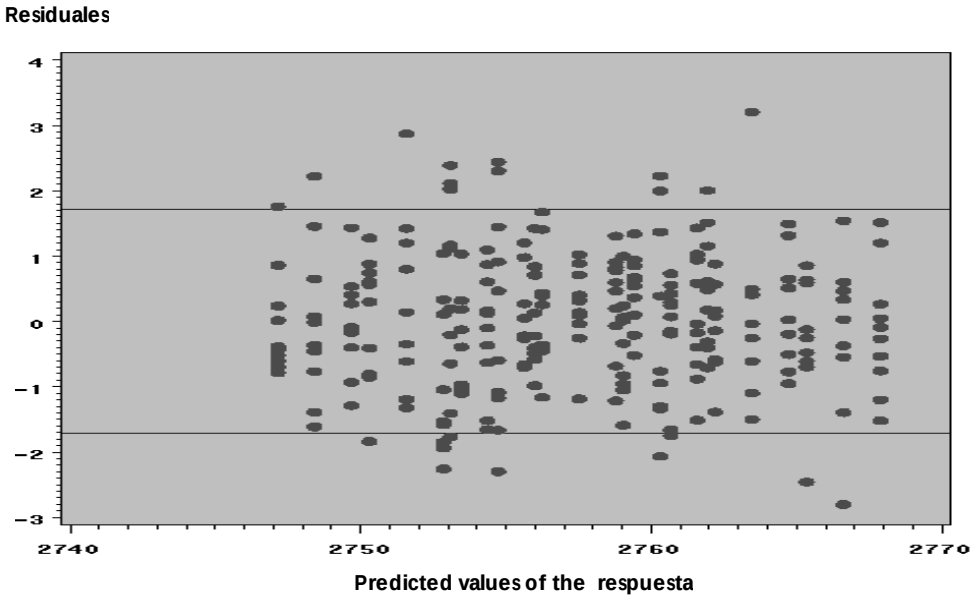


Figura 41. Gráfica de los residuales contra valores estimados de la respuesta para la instancia nug20

- **Instancia sko42**

La tabla ANOVA que arrojó el Proc REG para esta instancia fue la siguiente.

The REG Procedure
Model: regresion
Dependent Variable: respuesta

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	3	222231	74077	1.63	0.1838
Error	266	12122473	45573		
Corrected Total	269	12344704			

Root MSE	213.47882	**R-Square**	0.0180
Dependent Mean	18096	**Adj R-Sq**	0.0069
Coeff Var	1.17970		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	18109	56.63041	319.78	<.0001
evaporacion	1	5.58889	15.91177	0.35	0.7257
diversificacion	1	-29.90000	15.91177	-1.88	0.0613
exploracion	1	17.58889	15.91177	1.11	0.2700

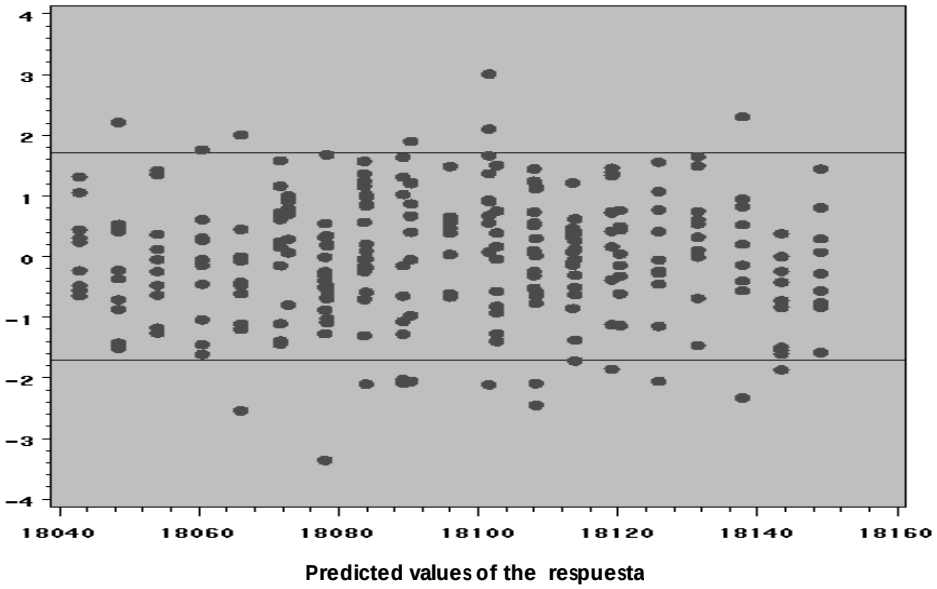


Figura 42. Gráfica de los residuales contra valores estimados de la respuesta para la instancia sko42

- **Instancia tai25a**

La tabla ANOVA que arrojó el Proc REG para esta instancia fue la siguiente.

The REG Procedure
Model: regresion
Dependent Variable: respuesta

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	3	848378817	282792939	1.50	0.2154
Error	266	50204484900	188738665		
Corrected Total	269	51052863717			

Root MSE	13738	**R-Square**	0.0166
Dependent Mean	1294670	**Adj R-Sq**	0.0055
Coeff Var	1.06114		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	1292916	3644.39429	354.77	<.0001
evaporacion	1	1038.62222	1023.98640	1.01	0.3114
diversificacion	1	-1426.32222	1023.98640	-1.39	0.1648
exploracion	1	1264.94444	1023.98640	1.24	0.2178

Residuales

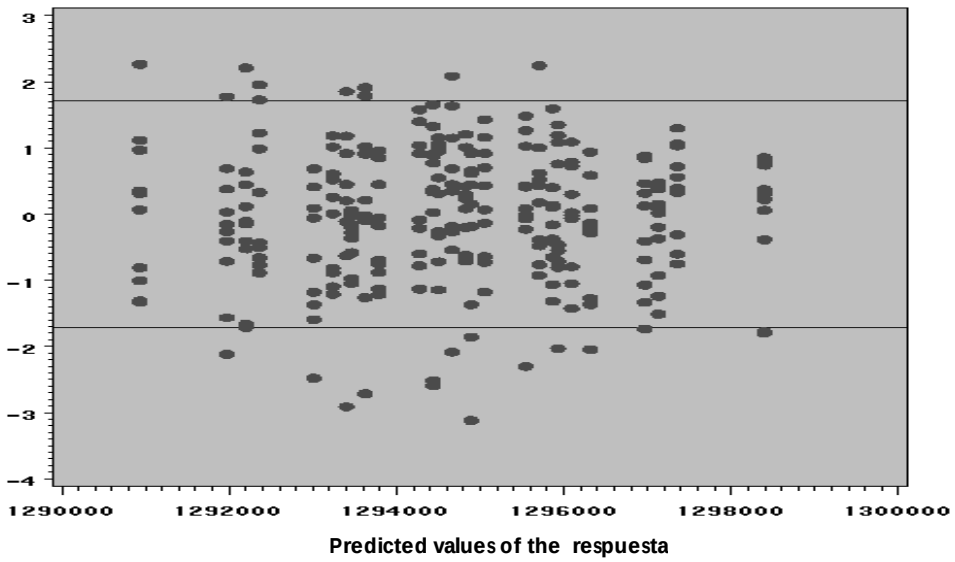


Figura 43. Gráfica de los residuales contra valores estimados de la respuesta para la instancia tai25a

Los resultados del análisis de regresión aplicando el Proc REG para cada una de las instancias fueron análogos a los obtenidos con el Proc GLM en la sección 7.7, como era esperado. De esta forma, el análisis de regresión sirvió para validar y complementar con otros estimados estadísticos, las conclusiones del diseño factorial. Se detalla a continuación la información suministrada por los estimados de las tablas anova.

- P-value de Model (`Pr > F`): Es una medida del ajuste del modelo de regresión a la explicación de la varianza de la respuesta. Corresponde también a la prueba de hipótesis Ho: $\beta 2 = \beta 3 = ... = \beta k = 0$ que determina la significancia simultanea de todos los mecanismos en la respuesta (en el diseño factorial esta prueba esta asociada al p-value de la interacción triple **evapor*divers*explor**). De acuerdo con esto, las instancia bur26e y chr25 fueron las únicas en donde esta medida fue significativa, lo que indica que la variación de sus respuestas podría estar explicada por el modelo de regresión planteado con los tres mecanismos, evaporación, diversificación y

exploración. En el resto de las instancias el modelo de regresión no se ajusta a la varianza de la respuesta y por ende la interacción conjunta de los tres mecanismos (interacción triple **evapor*divers*explor)** no influye significativamente en ella.

- R^2 (`R-Square`): Se interpreta como el porcentaje de la variabilidad de la respuesta que es explicado por el modelo de regresión. El porcentaje para todas las instancias es significativamente bajo. Lo anterior indica que aún aquellas instancias en las que el modelo de regresión parece ajustarse a la variación de la respuesta, el porcentaje explicado de esa variabilidad es muy pequeño. Así mismo evidencia que la relación entre la respuesta y los tres mecanismos no es totalmente lineal.

- P-values de los mecanismos (`Pr > |t|`): Son valores aproximados que miden la importancia independiente de un mecanismo en la respuesta. Sus valores para cada instancia son análogos a los obtenidos en el diseño factorial, esto significa que los mecanismos que resultaron ser significativos en la tabla ANDEVA para una determinada instancia, también lo son en la tabla ANOVA. De esa manera se reiteran las conclusiones dadas en el diseño factorial con respecto a la influencia de cada mecanismo por separado en la respuesta de las instancias.

- Estimados de los mecanismos (`Parameter Estimate`): De acuerdo al signo que tengan, determinan el incremento o decremento de la respuesta a partir del valor estimado del intercepto. En relación con el diseño factorial, son los que permiten clasificar los tratamientos dentro de los grupos definidos previamente con base a su desempeño medido en función del promedio de sus respuestas.
- Supuesto de normalidad de los residuales (Gráfica de los residuales): La representación gráfica de los residuales contra los valores estimados de la

respuesta es una de las herramientas más usadas en el análisis de regresión. Se empleó para validar el supuesto de homoscedasticadad de la varianza e identificar respuestas por fuera de la distribución. Por ser un modelo de regresión lineal simple no fue necesario verificar el supuesto de normalidad de los residuales. Las gráficas de la figura 37 a la 43 no mostraron ningún tipo de problema en los supuestos. Las respuestas están dispersas por encima y por debajo de la línea cero y muy pocas aparecen fuera de los límites.

Capítulo XII

MANUAL DEL USUARIO HAS-QAP-R

A continuación se dan los detalles para el uso del algoritmo implementado HAS-QAP-R.

12.1. SOFTWARE

- Abrir la plataforma Jbuilder 2005.
- Abrir el archivo HAS-QAP-R.
- En el archivo .txt **datos** ingresar los datos de la instancia en el orden que se da a continuación "con un solo espacio (**un click del botón enter**) entre cada uno de ellos". En este orden se encuentran en la librería QAPLIB que pueden ubicar en la página web http://www.qaplib.com:

    - *Tamaño de la instancia: n*
    - *Matriz de distancia: A*
    - *Matriz de flujos: B*

- Leer las instrucciones del archivo .txt **readme**.
- Abrir la plataforma de Java, Jbuilder 2005.
- Abrir el ejecutable e ingresar los datos de acuerdo al archivo readme.

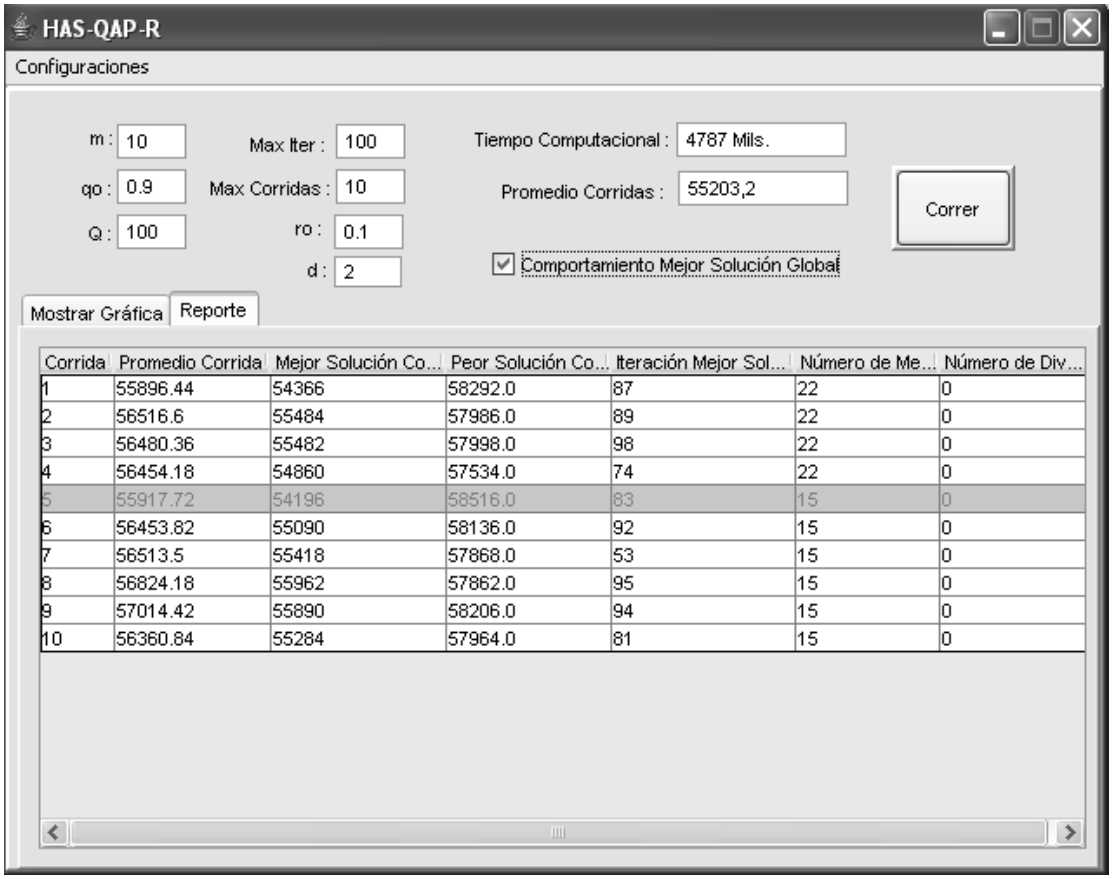
12.2. USO DEL SOFTWARE HAS-QAP-R

- Al abrir el ejecutable se abre el software HAS-QAP-R (Ventana HAS-QAP-R), en adelante ventana. Usted podrá ingresar los valores que considere en los siguientes parámetros de entrada:

- *Tamaño de la colonia de hormigas:* m
- *Parámetro para definir función de la regla pseudoproporcional aleatoria:* q_0
- *Parámetro para calcular el valor de* τ_0*:* Q
- *Número máximo de iteraciones:* Max Iter
- *Número máximo de corridas (replicas del algoritmo):* Max Corridas
- *Parámetro de evaporación:* ro
- *Parámetro para definir el valor de* S *(corresponde al denominador en la función* n/d *):* d

- Correr el programa HAS-QAP-R. Para ello hacer click en el botón **Correr**.
- Automáticamente se abre la gráfica HAS-QAP-R comentada en el capítulo IX, identificada por la pestaña "**Mostrar Gráfica**". Para observar el comportamiento de la Mejor Solución Global se debe seleccionar en la ventana la casilla "**Comportamiento Mejor Solución Global**". Así mismo se arrojan los valores del tiempo computacional empleado por el algoritmo y el promedio de las corridas identificados en la ventana por los siguientes nombres:

  - *Tiempo Computacional:* Corresponde al tiempo dado en milisegundos empleado por el algoritmo HAS-QAP-R.
  - *Promedio Corridas:* Corresponde al promedio de las mejores soluciones encontradas en cada corrida (*Mejor Solución Corrida*).
- El reporte de los resultados para la totalidad de las corridas se muestra en la ventana haciendo click en la pestaña "**Reporte**". La figura 44 que se observa más adelante es un ejemplo del reporte entregado por el programa HAS-QAP-R. El reporte contiene la siguiente información para cada réplica:



Corrida	Promedio Corrida	Mejor Solución Co...	Peor Solución Co...	Iteración Mejor Sol...	Número de Me...	Número de Div...
1	55896.44	54366	58292.0	87	22	0
2	56516.6	55484	57986.0	89	22	0
3	56480.36	55482	57998.0	98	22	0
4	56454.18	54860	57534.0	74	22	0
5	55917.72	54196	58516.0	83	15	0
6	56453.82	55090	58136.0	92	15	0
7	56513.5	55418	57868.0	53	15	0
8	56824.18	55962	57862.0	95	15	0
9	57014.42	55890	58206.0	94	15	0
10	56360.84	55284	57964.0	81	15	0

Figura 44. Ejemplo de un reporte entregado por el software HAS-QAP-R para la instancia sko 64 con los valores identificados en las casillas para cada uno de los parámetros.

- *Promedio Corrida:* Corresponde al promedio de las mejores soluciones encontradas en cada una de las iteraciones.
- *Mejor Solución Corrida:* Corresponde a la mejor solución encontrada por el algoritmo en el total de las iteraciones.
- *Peor Solución Corrida:* Corresponde a la peor solución encontrada en el total de las iteraciones.
- *Iteración Mejor Solución:* Corresponde a la iteración donde se obtuvo la *Mejor Solución Corrida*.
- *Número de mejoramientos:* Corresponde al número de veces que se consiguió actualizar (mejorar) la mejor solución global.

- *Número de diversificaciones:* Corresponde al número de veces que se aplicó el mecanismo de diversificación.

La **corrida resaltada** (en color rosado) en la figura 44 corresponde a la corrida que consiguió la mejor de todas las soluciones entre la totalidad de las corridas.

Capítulo XIII

RECOMENDACIONES

La influencia de los algoritmos de búsqueda local rápida 2-opt como el fast descent procedure en el desempeño que tienen los métodos HAS-QAP y MMAS-QAP$_{2\text{-opt}}$ en instancias de la categoría 1, incentivó el reemplazo del algoritmo de búsqueda tabú en GH por un fast descent procedure. El método generado fue el GDH propuesto en [25], con mejores resultados a los de GH en el mismo tiempo computacional.

La propuesta consiste en reemplazar el algoritmo fast descent procedure con el que funciona actualmente el algoritmo HAS-QAP^R, por un algoritmo de búsqueda tabú, y probar si su desempeño puede mejorar aún más en instancias de la categoría 2, y ver que tanto puede afectarse, resolviendo instancias de la categoría 1.

Los algoritmos GH y HAS-QAP son muy parecidos aunque pertenecen a metaheurísticas diferentes. Las similitudes que se encuentran son su inicialización a partir de soluciones generadas aleatoriamente, y la selección de las soluciones con las que empieza la próxima iteración. Esto sugiere el empleo de varios algoritmos de generación aleatoria que sirvan para definir su efecto sobre el desempeño de HAS-QAP^R y su contribución a mejorarlo en instancias de la categoría 1 para ponerse más al nivel de GH y HAS-QAP encontrando las soluciones óptimas.

Capítulo XIV

CONCLUSIONES

En este trabajo se implementa un algoritmo metaheurístico híbrido clasificado dentro de los algoritmos hormigas denominado HAS-QAP. Este algoritmo aunque posee varios de los mecanismos que caracterizan a los algoritmos ACO no puede ser considerado como tal al reemplazar el mecanismo de construcción de soluciones por uno de modificación empleado en cada iteración. Así mismo, utiliza uno de diversificación que lo reinicializa a partir de soluciones iniciales generadas aleatoriamente, con el fin de explorar nuevas regiones en la búsqueda de soluciones de mejor calidad a las encontradas hasta el momento. HAS-QAP es considerado como uno de los mejores algoritmos resolviendo instancias de la categoría 1, entre las que se incluyen instancias de la vida real (instancias de la clase (c)). Por esa razón se decidió implementar uno con el nombre de HAS-QAP^R. HAS-QAP^R usó los mismos parámetros propuestos para HAS-QAP que fueron considerados los más robustos para las instancias de la librería QAPLIB probadas en [18] y que fueron las mismas que se utilizaron para ponerlo a prueba frente a otros algoritmos mediante la aplicación de tres técnicas estadísticas multivariadas, anova, componentes principales y factores. Las técnicas mencionadas permitieron la identificación de los tipos de algoritmos que existen para resolver las dos categorías de instancias y descubrir la afinidad entre algoritmos del mismo tipo, así como la competencia de algunos para resolver ambas categorías. Los resultados de los tres análisis fueron los que se esperaban, de acuerdo a la clasificación que hacen de los algoritmos en la literatura y les da crédito como herramientas de análisis de datos en metaheurísticas.

HAS-QAP^R probó su habilidad como técnica metaheurística aplicada a problemas de asignación cuadrática, al obtener buenas soluciones en corridas cortas en tan sólo milisegundos, que superaban en calidad a las obtenidas por HAS-QAP, para varias de

las instancias de la vida real. Sin embargo, no supera la capacidad de HAS-QAP, y en ciertas instancias la de GH, para alcanzar las soluciones óptimas.

Las técnicas estadísticas lograron identificar los dos tipos de métodos que existen para resolver ambas categorías. En el primero se confirman los métodos HAS-QAP y GH, y se une a ellos el HAS-QAP^R. En el segundo están los métodos TS, RTS y SA. Dentro del primero, la afinidad esta entre HAS-QAP y GH; en el segundo, esta entre TS y RTS, determinada por el uso de una búsqueda tabú como algoritmo de búsqueda local. En instancias de la categoría 1 el algoritmo TS demostró tener el peor desempeño, mientras que el algoritmo SA en corridas largas pudo colocarse casi al nivel de los algoritmos del primer tipo. En instancias de la categoría 2 es más evidente la diferencia entre las clases que la conforman. De acuerdo con los resultados del análisis de componentes principales las instancias de la clase (a) son más fáciles de resolver en promedio por todos los métodos, con soluciones muy superiores en calidad a las obtenidas para las instancias de la clase (b). Para las instancias de la categoría 2 el mejor desempeño lo tienen los métodos HAS-QAP^R, TS y GH, seguidos por HAS-QAP y SA; el método RTS demostró tenerlo para las instancias de la clase (a), con mejores resultados a los de TS en casi todas ellas. Con lo anterior se deja en claro la influencia del tipo de algoritmo de búsqueda local utilizado, en el desempeño de los algoritmos dentro de cada tipo de instancia. Métodos con búsqueda tabú como algoritmo de búsqueda local, están más dirigidos a instancias de la categoría 2 (TS y RTS), mientras que aquellos que utilizan algoritmos de mejora iterativa, como el HAS-QAP y el HAS-QAP^R, lo están hacia las de la categoría 1.

La separación del método HAS-QAP^R de sus compañeros del tipo 1, los métodos GH y HAS-QAP, en la solución de las instancias, se alcanza a ver con el análisis de factores de la sección 7.5. Esta separación identifica una estructura para HAS-QAP^R que comparte características de las estructuras de los dos tipos de algoritmos, que explica su buen desempeño tanto en las instancias de la categoría 1 como en las de la

categoría 2. Encontramos la misma habilidad para resolver ambas instancias en el algoritmo GH, que es un híbrido que utiliza como algoritmo de búsqueda local una búsqueda tabú, mientras que HAS-QAP^R utiliza uno de mejora iterativa como lo es el fast descent procedure.

Por último, un diseño factorial tomando los mecanismos característicos de HAS-QAP^R, evaporación, diversificación y exploración mostró que la respuesta entregada por el algoritmo no depende de los niveles que tomen, y que para cualquier combinación de estos se obtienen respuestas de buena calidad. Esta conclusión fue validada por un análisis de varianza que evaluó el ajuste de un modelo de regresión lineal y los supuestos de los residuales a través de su representación gráfica contra los valores estimados de la respuesta.

El algoritmo HAS-QAP como tema de estudio ya lleva varios años en países desarrollados, en Colombia no esta exento de investigación y por ende de su aplicación como herramienta logística de optimización. Por ese motivo, espero que este trabajo sea útil como material de estudio y de incentivo para próximas investigaciones.