M-ACO, un Algoritmo de Multicolonias de Hormigas para el Problema de Múltiples Ascensores

S. Molina[1], G. Leguizamón[1] and E. Alba[2]

[1] Universidad Nacional de San Luis, Argentina
[2] Universidad de Málaga, España

Resumen El *Problema de Múltiples Ascensores* (MEP) es un problema de optimización combinatorio, dinámico no estacionario, consiste en encontrar la secuencia de movimientos para cada ascensor de un edificio de manera tal de minimizar el tiempo medio de espera de los pasajeros. En este trabajo se propone un algoritmo ACO *Multicolonia de Hormigas* (M-ACO) para el MEP que implementa una estrategia simple para adaptarse a los cambios basada en la modificación de los rastros de feromona. Se aplican y analizan métricas específicas para entornos dinámicos y paralelos para medir el desempeño del algoritmo.

Palabras Claves
Problema de Múltiples Ascensores, Problemas No Estacionarios, ACO Multicolonia de Hormigas.

1. Introducción

La metaheurística *Optimización de Colonias de Hormigas* (ACO, *Ant Colony Optimization*) [4], se inspira en el comportamiento de las colonias de hormigas naturales. Es aplicable a problemas de optimización combinatorios (POC) duros en ambientes estacionarios y no estacionarios, estáticos y dinámicos.

En particular, un algoritmo ACO *Multicononias de hormigas* [1,2,16,17], posee varias colonias de hormigas cada una con su propia matriz de feromonas las cuales, para la construcción de una solución, cooperan con sus colonias vecinas, con algún tipo de información. En un entorno paralelo, las colonias de hormigas son distribuidas en los procesadores disponibles, utilizando así la potencia de cómputo y los recursos que brindan hoy en día las arquitecturas paralelas y las redes de computadoras.

El *Problema de Múltiples Ascensores* (MEP) [13,14,18,20] es un problema de optimización combinatorio dinámico (POD) [12,15] no estacionario, que consiste en encontrar una secuencia de movimientos que deben realizar los ascensores de un edificio de manera tal de minimizar el tiempo medio de espera de los pasajeros. La llegada de un nuevo pasajero a la cola de un ascensor, la rotura de un ascensor, etc. son eventos que provocan cambios de estado en el problema,

haciéndolo un problema dinámico ya que se modifica la instancia del problema para la cual se intenta encontrar una solución.

En este trabajo se propone un algoritmo ACO *Multicolonia de Hormigas* (M-ACO) para el MEP, se aplica una estrategia especial de modificación de la matriz de feromonas cuando ocurre un cambio, la cual es introducida con el propósito de lograr una mejor adaptación del algoritmo a los cambios del entorno del problema. Además, se muestra la aplicabilidad de las métricas específicas para entornos dinámicos y paralelos para medir el desempeño del algoritmo en cuanto a la calidad de las soluciones obtenidas, el costo de adaptación del mismo y el tiempo de ejecución.

2. El Problema de Múltiples Ascensores

En términos generales, el *Problema de los Múltiples Ascensores* (MEP) consiste de un edificio de varias plantas con un grupo de ascensores, en donde los usuarios generan llamadas a los mismos para trasladarse de una planta a otra.

Cada ascensor debe realizar un recorrido eficiente de las plantas de manera tal que se minimice por ejemplo, el tiempo medio de espera de los usuarios, el tiempo medio de tránsito de los usuarios, la cantidad de usuarios dentro de un ascensor o el consumo de energía.

Para el MEP se han realizado investigaciones con el objetivo de estudiar el comportamiento de un grupo de ascensores, a través de simuladores, probando diferentes políticas de planificación y técnicas algorítmicas para encontrar la secuencia de movimientos de los ascensores. Por ejemplo: Mateus Rosso y Méndez [13] implementan un simulador en JAVA que permite configurar las características del edificio e incluir nuevas políticas de planificación; Molina et al. [14] implementan un simulador en C++ el cual se incluye en un algoritmo ACS para encontrar la solución al Problema de Unico Ascensor; Yu et al. [20] prueban un algoritmo híbrido, el cual utiliza Programación de Redes Genéticas y Optimización de Colonias de Hormigas para un sistema de control y supervición de un grupo de ascensores; Márquez Torres [18] incluye un simulador que interactúa con un algoritmo genético; Ho y Robertson [8] aplican Redes de Petri combinadas con Redes Neuronales para encontrar la secuencia de movimientos que debe realizar un ascensor; Eguchi et al. [5] aplican Programación Red-Genética para evolucionar un controlador de un sistema de ascensores múltiples. Otros aplican enfoques alternativos para abordar el problema, como controladores difusos, Sistemas Expertos, Redes Neuronales y combinación de los mismos [3,7,9,10].

2.1. Formulación del Problema

Como se muestra en la figura 1, el MEP se puede definir en términos de las siguientes componentes: (i) el conjunto de las n plantas $P = \{p_1, p_2, \ldots, p_n\}$; (ii) el conjunto de los m ascensores del edificio $A = \{A_1, A_2, \ldots, A_m\}$, las velocidades y capacidades de cada ascensor, $\overrightarrow{V} = \langle v_1, v_2, \ldots, v_m \rangle$ y $\overrightarrow{C} = \langle c_1, c_2, \ldots, c_m \rangle$

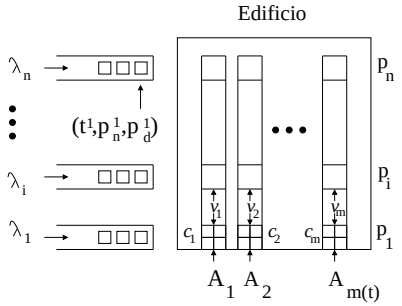

Figura 1. Esquema general del MEP. La llamada (t^l, p_n^l, p_d^l) se genera en el tiempo t^l desde la planta p_o^l y hacia la planta p_d^l. λ_i son las velocidades medias de arribos de la llamadas para una determinada planta i.

respectivamente. La función $m(t) : t \to \mathbb{N}$, retorna el número de ascensores disponibles en un tiempo determinado y (iii) las funciones de distribución Poisson f_{λ_i}, con parámetro λ_i para $i \in \{1, \ldots, n\}$. Este parámetro respresenta la velocidad media de generación de llamadas por unidad de tiempo en la planta i, puede variar respecto a las plantas y respecto al tiempo (t); por lo tanto se tiene el vector de funciones de distribución para cada planta, para un determinado tiempo: $\overrightarrow{f}_\lambda(t) = \langle f_{\lambda_1}, f_{\lambda_2}, \ldots, f_{\lambda_n} \rangle$, con $\lambda_i(t) : t \to \mathbb{N}$.

2.2. Instancias del Problema

Para los estudios realizados en este trabajo, el número de ascensores no cambia con el tiempo, los ascensores tienen la misma velocidad y capacidad. Por lo tanto reemplazamos las componentes $m(t)$, $\overrightarrow{V}$ y $\overrightarrow{C}$ por las constantes m, v y c respectivamente.

Aquí, consideramos una única fuente de dinamismo usada para el MEP es la secuencia de llamadas $L(\overrightarrow{f_\lambda}(t)) = \langle (t^1, p_o^1, p_d^1), \ldots, (t^l, p_o^l, p_d^l) \rangle$ (en donde t^1 es el tiempo de ocurrencia de la llamada, p_o^l la planta origen de la llamada, p_d^l la planta destino de la llamada) debido a la variación en el tiempo de las funciones f_λ. De acuerdo a estas características, la frecuencia de cambio de L es discreta, es decir, la instancia del problema cambia cada cierto periodo de tiempo. Así, el MEP puede ser visto como una secuencia de problemas estáticos diferentes. Ésta es una de las posibles maneras de definir problemas dinámicos (Leguizamón et al. [12]). Por lo tanto, una instancia queda definida como $I = \langle n, m, v, c, \delta_C \rangle$ con $\delta_C = (\overbrace{L_1, L_2, L_3}^{ciclo_1}, \ldots, \overbrace{L_1, L_2, L_3}^{ciclo_C})$, el subíndice $C \in \mathbb{N}$, indica el número de ciclos de la forma (L_1, L_2, L_3). El período de tiempo que se considera una determinada secuencia L_i de δ_C, se denomina *Etapa* y su tiempo de duración es denotado con t_e. Por ejemplo, con δ_2 y $t_e = 50$, indica que se producen dos ciclos de secuencias de llamadas, resultando $\delta_2 = (L_1, L_2, L_3, L_1, L_2, L_3)$ y que L_1 será considerada 50 unidades de tiempo antes de que ocurra un cambio de secuencia, siguiendo L_2 y así sucesivamente.

2.3. Evaluación de la Calidad de una Solución del Problema

Una solución para el MEP consiste de una secuencia de plantas a visitar para cada ascensor (basada en el modelo del SEP propuesto en Molina et al. [14]). Por ejemplo, la figura 2 muestra una secuencia posible de visitas de longitud $k_1 = 6$ y $k_2 = 8$ para los ascensores A_1 y A_2 respectivamente, en un edificio con dos ascensores ($m = 2$).

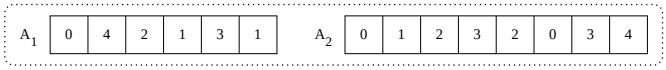

Figura 2. Secuencias para el MEP, de longitud $k_1 = 6$ para A_1 y $k_2 = 8$ para A_2.

La función de evaluación de la calidad de una solución se basa en la función propuesta en Márquez Torres [18] y a su vez se aplica en Molina et al. [14] y su formulación está dada por: $f = \frac{1}{n}\sum_{i=1}^{n}\mu_i$ con $\mu_i = \frac{1}{\beta_i}\sum_{j=1}^{\beta_i}\mu_{ij}$, en donde μ_i es el tiempo medio de espera total de las personas en la planta i; μ_{ij} es el tiempo medio de espera de una persona j en la planta i; y β_i es el número de personas que han esperado en la planta i.

3. Algoritmo M-ACO

En el algoritmo M-ACO (algoritmo 1) varias colonias de hormigas, exploran el espacio de búsqueda utilizando su propia matriz de feromonas y cooperan entre sí con algún tipo de información con sus colonias vecinas. En este trabajo nos basamos en el modelo ACO presentado en Molina et al. [14], adoptando así, el *Grafo de Construcción Parcialmente Conectado*, la *Regla Proporcional Pseudo-random* y la información heurística: la distancia entre dos plantas, la longitud media de cola, el tiempo medio de espera y el nro. de pasajeros dentro del ascensor. Este algoritmo, interactúa con el algoritmo *simulador MEP*, el cual simula los movimientos de los ascensores según lo que indica la solución en evaluación; y con el algoritmo $Admin\_I_d$ el cual administra las instancias dinámicas del problema en función del tiempo de simulación (controla los cambios de secuencias según δ_C). En segundo lugar, cada colonia de hormigas (C_i) inicializa la matriz de feromonas (paso 3). Luego, en forma repetitiva en cada colonia: se inicializan las estructuras de datos según los valores paramétricos leídos (paso 5), cada hormiga (k) de una colonia construye una solución (paso 6), el algoritmo M-ACO envía al simulador del MEP los próximos movimientos que debe realizar cada ascensor, para que los simule. Luego, el simulador retorna a M-ACO el nuevo estado del edificio. El algoritmo $Admin\_I_d$ le proporciona al simulador la instancia I_d a simular en un determinado momento y le avisa al algoritmo M-ACO si ha ocurrido un cambio. Esto no implica que M-ACO tenga conocimiento de tipo de cambio o que sepa de antemano cuándo ocurre el cambio. El aviso es una simple señal de cambio para que el algoritmo aplique la estrategia de adecuación de la

Algorithm 1 M-ACO

```
1: BEGIN
2: LecturaDeParámetros();
3: InicializarMatrizDeFeromonas_{C_i}();
4: for (iter = 1; iter <= Max_iter; iter++) do
5:    InicializarColonia_{C_i}();
6:    (∀k ∈ C_i) ConstruirUnaSolución();
7:    MejorSoluciónLocal_{C_i}=ElegirMejorSoluciónLocal();
8:    MejorSoluciónGlobal_{C_i}=ActualizarSoluciónGlobal();
9:    Evaporar();
10:   if (PasoDeComunicación) then
11:      EnviarSolución(ColoniaVecina,MejorSoluciónGlobal_{C_i});
12:      MejorSoluciónVecina=RecibirSolución(ColoniaVecina);
13:      Intensificar(MejorSoluciónVecina);
14:   end if
15:   Intensificar(MejorSoluciónLocal_{C_i}, MejorSoluciónGlobal_{C_i});
16:   if (Hay_cambios()) modificar_τ(γ); end if
17: end for
18: if (C_i ≠ C_main) then
19:    EnviarSolución(MejorSoluciónGlobal_{C_i},C_main);
20: else
21:    Solución=RecibirSolución(C_i);
22:    MejorSoluciónGlobal_{C_main}=ActualizarSoluciónGlobal();
23: end ifEND
```

matriz de feromonas. Luego, se elige la mejor solución de todas como la *Mejor Solución Local*, en caso de ser necesario la *Mejor solución Global* es reemplazada por ésta (pasos 7 y 8). Luego, se realiza la evaporación (paso 9) seguida de un proceso de comunicación basada en un modelo sincrónico, durante el cual, migra la *Mejor Solución Global* (pasos 11 y 12), se aplica la topología de comunicación *Anillo*. Acontinuación, se realiza un proceso de intensificación de los rastros de feromona usando la *Mejor Solución Local*, la *Mejor Solución Global* y la *Solución Vecina* (pasos 13 y 15). Seguidamente, si ha ocurrido un cambio, se actualizan todos los valores $\tau_{i,j}$ de la matriz de feromonas utilizando la función (Leguizamón y Alba [11]): $\tau_{i,j}^{new} = (1-\gamma)\tau_{i,j}^{old} + \gamma\tau_0$ (paso 16). Del *Factor de actualización o adecuación* γ depende la cantidad de experiencia adquirida que preserva el algoritmo luego de un cambio, y por ende el grado de exploración del mismo. Por último, se elige (en la colonia C_{main}) la mejor de todas las soluciones producidas por cada colonia, que es la solución final que retorna el algoritmo.

4. Marco Experimental

El marco experimental presentado, tiene el objetivo de estudiar el desempeño del algoritmo M-ACO, variando el número de procesadores utilizados, respecto a: (i) la calidad de las soluciones obtenidas, analizando los TME^* (valores objetivos promedios a lo largo de las etapas) y utilizando las métricas *Exactitud*

y *Error*; (ii) la adaptación del algoritmo a los cambios del problema, utilizando la métrica *Estabilidad* y *Reactividad-*ϵ (Leguizamón et al. [12], Feng et al. [6], Weiker [19]); y (iii) los tiempos de ejecución obtenidos utilizando las métricas *Speedup*, *Eficiencia* y *Eficacia*. Se utilizan los valores de referencias (V_{ref}), los mejores valores objetivos conocidos para cada una de las etapas, y calculados previamente.

Experimento	I_d	n	m	v	c	δ_C	t_e	γ
$E_{(I_{d_1},\gamma)}$	I_{d_1}	25	2	2.6	15	δ_8	50	{0.2,0.5,0.8,0}
$E_{(I_{d_2},\gamma)}$	I_{d_2}	25	2	2.6	15	δ_4	100	{0.2,0.5,0.8,0}
$E_{(I_{d_3},\gamma)}$	I_{d_3}	25	2	2.6	15	δ_2	200	{0.2,0.5,0.8,0}
$E_{(I_{d_4},\gamma)}$	I_{d_4}	25	4	2.6	15	δ_8	50	{0.2,0.5,0.8,0}
$E_{(I_{d_5},\gamma)}$	I_{d_5}	25	4	2.6	15	δ_4	100	{0.2,0.5,0.8,0}
$E_{(I_{d_6},\gamma)}$	I_{d_6}	25	4	2.6	15	δ_2	200	{0.2,0.5,0.8,0}

Cuadro 1. Configuraciones de los experimentos realizados.

El cuadro 1 muestra los experimentos realizados, incluye la descripción de las instancias I_d usadas y los valores del Factor de actualización (γ) estudiados. Para cada experimento se calcularon las métricas en función de los valores medios obtenidos al considerar los resultados de 30 ejecuciones independientes.

Los valores paramétricos del M-ACO, considerados fueron: Max_iter=1200, nro. de iteraciones entre comunicaciones $iter_i = 10$, $k = 20$, $\rho = 0.5$, $q_0 = 0.8$, $\beta = 2$, $\tau_0 = 0.8$, los factores de la heurística local η: $\alpha_1 = 0.05$, $\alpha_2 = 0.3$, $\alpha_3 = 0.7$ y $\alpha_4 = 0.5$.

4.1. Análisis de los Resultados

Para el análisis de los resultados se consideran los aspectos (i)-(iii) de la sección 4, se aplica el test no paramétrico de *Kruskal Wallis* para comparar los resultados obtenidos en cada experimento y concluir así sobre la significancia de las diferencias entre éstos.

Experimento	**E1-(0.136)**	**E2-(0.179)**	**E3-(0.213)**	**E4-(0.136)**	**E5-(0.179)**	**E6-(0.213)**
$E_{(I_{d_3},0.2)}$	0.152	0.201	**0.230**	0.161	0.200	0.212
$E_{(I_{d_3},0.0)}$	0.157	0.216	**0.279**	0.177	0.217	0.256

Cuadro 2. Valores TME^* para cada etapa (E1-E6) para los experimentos $E_{(I_{d_3},0.2)}$, $E_{(I_{d_3},0.0)}$ con 2 procesadores. Los valores V_{ref} se asocian a cada etapa. En E3 las diferencias son más extremas entre los TME^* alcanzados para cada experimento.

(i) Calidad de las Soluciones Obtenidas. El test estadítico *Kruskal Wallis*, aplicado para concluir sobre la significancia de las diferencias entre los TME^* al variar el factor γ, indica para cada tipo de experimento ($E_{(I_{d_1},\gamma)}$ - $E_{(I_{d_6},\gamma)}$) que

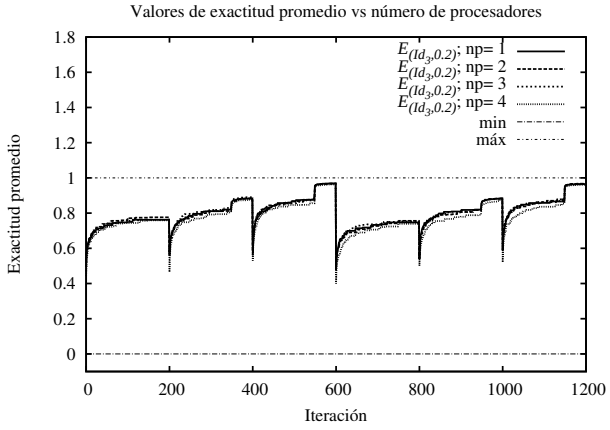

Figura 3. Exactitud promedio para el experimento $E_{(I_{d_3},0.2)}$ variando el número de procesadores (np). El valor mínimo (min) y máximo (máx) posible de Exactitud es 0 y 1 respectivamente.

las diferencias de los resultados son significativas entre aplicar la estrategia de modificación de feromonas ($\gamma = 0.2, 0.5$ y 0.8) en lugar de no aplicarla ($\gamma = 0.0$), ya que los valores p resultantes, son menores al nivel de significancia 0.05. Esta diferencia es más notable a partir de la etapa 3 (E3), cuando el algoritmo debe encontrar una solución a una situación más compleja, en donde se considera la secuencia de llamadas L_3 (con distribución Poisson con $\lambda = 30$). Además, de haber adquirido cierta experiencia de búsqueda, al finalizar E3 el cambio es más severo ya que la nueva etapa comienza con la secuencia L_1 (con distribución Poisson con $\lambda = 10$). Por ejemplo, en el cuadro 2 se muestran los TME^* para las etapas (E1-E6) y sus respectivos V_{ref} (junto al nombre de la etapa), de los experimentos $E_{(I_{d_3},0.2)}$ y $E_{(I_{d_3},0.0)}$, con un número de procesadores (np) igual a 2 en donde se observa lo mencionado anteriormente, en E3 se muestran las diferencias más extremas entre los TME^* alcanzados para cada experimento.

En cambio, las diferencias de los TME^* no son significativas cuando se aplica la estrategia de modificación de feromona con distintos valores de $\gamma > 0.0$ (los valores p resultantes de aplicar el test son mayores a 0.05).

La figura 3 muestra los valores de exactitud variando el número de procesadores (np) para el $E_{(I_{d_3},0.2)}$, para cada una de las etapas los valores de *Exactitud* más cercanos a uno se encuentran al final de las mismas. En particular, en el primer ciclo ($E1 - E3$, primeras 600 iteraciones) la exactitud final en cada etapa va creciendo en relación al valor alcanzado en la etapa anterior. Esto muestra que la experiencia adquirida es útil dentro de una misma etapa como así también para las próximas etapas, a pesar de que las nuevas secuencias provocan cambios cada vez más severos (dados por el aumento en el número de llamadas por unidad de tiempo). En el segundo ciclo ($E4 - E6$, últimas 600 iteraciones) la exactitud disminuye notablemente cuando se cambia de la E3 a E4, en donde E4 utiliza una secuencia menos compleja que en E3.

Los valores observables al inicio de cada etapa, muestran la sensibilidad del algoritmo a los cambios de la instancia del problema ya que la exactitud disminuye abruptamente (el Error aumenta). En las etapas de los ciclos que utilizan la misma secuencia de llamadas (por ejemplo E1 y E4) el *Error* aumenta pero no de manera significativa.

(ii) Costo de Adaptación a los Cambios. El algoritmo es altamente estable para cada una de las etapas, ya que los valores de *Estabilidad* son cercanos a cero (aunque con tendencia a subir al inicio de cada una de las etapas). El mayor valor de estabilidad se alcanza en la etapa E3 indicando que el algoritmo tiene mayor dificultad de adaptación ante cambios más severos. El cuadro 3 muestra los valores de las diferentes métricas, para el experimento $E_{(I_{d_3},0.2)}$, se resaltan los valores extremos alcanzados, se puede observar lo mencionado anteriormente. Un punto interesante de comentar para la métrica *Reactividad*$-\epsilon$ (la cual mide

Métrica	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**
Exactitud	0.7350	0.8120	**0.8820**	0.7100	0.8160	**0.8730**
Error	0.1520	0.1240	**0.0870**	0.1750	0.1230	**0.0840**
Estabilidad	0.0011	0.0014	**0.0025**	0.0009	0.0015	**0.0000**

Cuadro 3. *Exactitud*, *Error* y *Estabilidad* promedios para el experimento $E_{(I_{d_3},0.2)}$ para cada etapa (E1-E6) para $np = 2$.

np	**E1**($t = 200$)	**E2**($t = 400$)	**E3**($t = 600$)	**E4**($t = 800$)	**E5**($t = 1000$)
1	12	49	548	9	53
2	14	37	548	10	41
3	19	54	548	10	59
4	18	93	548	11	86

Cuadro 4. Valores de Reactividad$-\epsilon$ para los experimentos $E_{(I_{d_3},0.2)}$, para la última unidad de tiempo (t), variando el nro. de procesadores (np).

np	**Speedup**	**Eficiencia**	**Eficacia**
2	**1.739**	**0.869**	**1.511**
3	2.047	0.682	1.397
4	2.720	0.680	1.850

Cuadro 5. Speedup, Eficiencia y Eficacia para el experimento $E_{(I_{d_3},0.2)}$ obtenidos al variar el nro. de procesadores (np).

cuántas unidades de tiempo tiene que transcurrir después del cambio hasta alcanzar una exactitud similar a la lograda antes del cambio), es respecto a lo que ocurre al final de cada etapa, ya que para el resto de las unidades de tiempo (t) independientemente de la etapa el valor es de 1 (indicando una rápida adaptación del algoritmo), en cambio en los tiempos $t = 200, 400, 600, 800$ y 1000 se observan valores mayores a 1. En particular para $t = 600$ se alcanza el mayor valor, indicando que al algoritmo le cuesta adaptarse a los cambios mucho más cuando pasa de considerar una secuencia L_3 a considerar una secuencia L_1. Por ejemplo, el cuadro 4 muestra los valores de reactividad para el experimento $E_{(I_{d_3},0.2)}$ variando el número de procesadores, se puede observar

que cuando $t = 600$ el algoritmo alcanza un valor de exactitud similar después de 548 unidades de tiempo (independientemente del número de procesadores), cuando nuevamente se considera L_3 en E6 y habiendo transcurrido 148 unidades de tiempo dentro de esta etapa.

Respecto a la influencia en la calidad de las soluciones, al variar el número de procesadores, se observa que no existen diferencias significativas en los TME^* obtenidos cuando el número de procesadores aumenta.

(iii) Tiempo de Ejecución. Los valores medios de los tiempos de ejecución, el *Speedup*, la *Eficiencia* y la *Eficacia* obenidos para cada experimento, difieren significativamente respecto a los valores obtenidos al variar el *np*. El speedup aumenta cuando el número de procesadores crece, en particular para $np = 2$ se alcanzan valores más cercanos al Speedup Ideal (2, 3 y 4 para $np = 2, 3$ y 4 respectivamente), lo comentado se refleja al observar los valores de la eficacia y la eficiencia (ver cuadro 5). Esto se debe a la sobrecarga generada por las comunicaciones entre los procesos proveniente de la comunicación entre las colonias de hormigas durante el proceso de migración y la sincronización proveniente del modelo sincrónico elegido para la comunicación.

5. Conclusiones

En este trabajo presentamos un algoritmo ACO *Multicolonias de Hormigas*, basado en un modelo sincrónico para el MEP. Hemos realizado varios experimentos con el objetivo de medir el desempeño del algoritmo en cuanto a la calidad de las soluciones obtenidas, el costo de adaptación a los cambios y el tiempo de ejecución.

Se logró mejorar el desempeño del algoritmo al aplicar una técnica de modificación de la matriz de feromonas ante los cambios de entorno del problema.

El algoritmo se adapta rápidamente a los cambios en la mayoría de los casos, excepto en las etapas más complejas.

Los tiempos de ejecución mejoran cuando el número de procesadores aumenta. Por lo tanto, se considera interesante seguir investigando sobre las características de la comunicación entre las colonias durante el proceso de migración y diferentes tipos de instancias de estudio del problema.

Agradecimientos

Los dos primeros autores agradecen el apoyo de la UNSL a través del proyecto de investigación P38403. El tercer autor agradece el apoyo financiero del CICE de la Junta de Andalucía, contrato P07-TIC-03044 (DIRICOM http://diricom.lcc.uma.es) y el Ministerio Español de Ciencias e Innovación (MICINN) y FEDER, contrato TIN2011-28194 (RoadMe http://roadme.lcc.uma.es).

Referencias

1. E. Alba. *Parallel Metaheuristics: A New Class of Algorithms.* Wiley-Interscience, 2005.

2. V. Cung, S. L. Martins, C. C. Ribeiro, and C. Roucairol. Strategies for the Parallel Implementation of Metaheuristics. In *Essays and Surveys in Metaheuristics*, pages 263–308. Kluwer Academic Publishers, 2002.
3. Z. Dewen, J. Li, Z. Yuwen, S. Guanghui, and H. Kai. Modern Elevator Group Supervisory Control Systems and Neural Networks Technique. In *IEEE International Conference on Intelligent Processing Systems, ICIPS '97.*, pages 528–532, 28-31 Oct. 1997.
4. M. Dorigo, G. Di Caro, and L. M. Gambardella. *The Ant Colony Optimization Metaheuristic*, pages 11–32. McGrawHill, London, 1999.
5. T. Eguchi, K. Hirasawa, J. Hu, and S. Markon. Elevator Group Supervisory Control Systems Using Genetic Network Programming. In *Congress on Evolutionary Computation, 2004. CEC2004*, volume 2, pages 1661–1667, 19-23 Jun. 2004.
6. W. Feng, T. Brune, L. Chan, M. Chowdhury, C. K. Kuek, and Y. Li. Benchmarks for testing evolutionary algorithms. Technical Report CSC-97006, University of Glasgow, Center for System and Control, Glasgow, UK, 1997.
7. R. Gudwin, F. Gomide, and M. Netto Andrade. A Fuzzy Elevator Group Controller with Linear Context Adaptation. In *IEEE International Conference on Fuzzy Systems Proceedings. IEEE World Congress on Computational Intelligence*, pages 481–486, 4-9 May. 1998.
8. M. Ho and B. Robertson. Elevator Group Supervisory Control Using Fuzzy Logic. volume 2, pages 825–828, Halifax, NS, 25-28 Sep. 1994. Electrical and Computer Engineering.
9. C. B. Kim, K. A. Seong, L. K. Hyung, and J. O. Kim. Design and Implementation of a Fuzzy Elevator Group Control System. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 28(3):277–287, May. 1998.
10. C. B. Kim, K. A. Seong, L. K. Hyung, J. O. Kim, and Y. B. Lim. A Fuzzy Approach to Elevator Group Control System. *IEEE Transactions on Systems, Man and Cybernetics*, 25(6):985–990, Jun. 1995.
11. G. Leguizamón and E. Alba. *Ant Colony Based Algorithms for Dynamic Problems*, chapter Metaheuristics for Dynamics Problems. Springer-Verlag, 2012. In Press.
12. G. Leguizamón, G. Ordóñez, and S. Molina. *Optimization Techniques for Solvin Complex Problems*, chapter Canonical Metaheuristics for Dynamic Optimization Problems, pages 83–96. John Wiley & Sons, Inc., Hoboken, New Jersey, 2009.
13. E. A. Mateus Rosso and S. J. J. Méndez. Desarrollo de un Simulador de un Edificio con Parámetro Variable. *Dialnet*, 19:107–120, Jun. 2009.
14. S. Molina, G. Leguizamón, and E. Alba. An ACO Model for a Non-stationary Formulation of the Single Elevator Problem. *JCS&T*, 7(1):45–51, 1 Apr. 2007.
15. T. T. Nguyen, S. Yang, and J. Branke. Evolutionary Dynamic Optimization: A Survey of the State of the Art. *Evolutionary Computation*, pages 1–24, May. 2012.
16. M. Pedemonte, S. Nesmachnow, and H. Cancela. A Survey on Parallel Ant Colony Optimization. *Applied Soft Computing*, 11(8):5181–5197, 2011.
17. M. Randall and A. Lewis. A Parallel Implementation of Ant Colony Optimization. *J. Parallel Distrib. Comput.*, 62(9):1421–1432, 2002.
18. R. Márquez Torres. Algoritmos Evolutivos Distribuidos en Entornos Dinámicos. Master's thesis, Departamento Lenguajes y Ciencias de la Computación. Universidad de Málaga, 2000.
19. K. Weicker. Performance measures for dynamic environments. In *Parallel Problem Solving from Nature–PPSN VII*, pages 64–73. Springer–Verlag, 2002.
20. L. Yu, J. Zhou, S. Mabu, K. Hirasawa, J. Hu, and S. Markon. Doubledeck Elevator Group Supervisory Control System Using Genetic Network Programming with Ant Colony Optimization with Evaporation. *JACIII*, 11(9):1149–1158, 2007.