

# Hibridación entre Filtro de Partículas y Búsqueda Dispersa: aplicación al TSP Dinámico

Juan José Pantrigo    Abraham Duarte    Ángel Sánchez

juanjose.pantrigo@urjc.es    abraham.duarte@urjc.es    angel.sanchez@urjc.es

Dto. Inform., Estad. y Telem.

Univ. Rey Juan Carlos

ESCET

28933 Móstoles

## Resumen

Este trabajo presenta el algoritmo de filtro de partículas con búsqueda dispersa (*Scatter Search Particle Filter - SSPF*) y su aplicación al problema del TSP dinámico. El algoritmo SSPF combina métodos de estimación secuencial (filtro de partículas) y de optimización combinatoria (búsqueda dispersa) para mejorar el rendimiento de la búsqueda dispersa en problemas de optimización dinámica. La estrategia del SSPF consiste en emplear las buenas soluciones en instantes previos para construir nuevas soluciones iniciales de alta calidad en instantes sucesivos, sin necesidad de restablecer el cómputo desde el inicio. Los algoritmos desarrollados se han probado sobre versiones dinámicas de instancias del banco de pruebas TSPLIB. Los resultados experimentales muestran que el rendimiento del SSPF es mayor que las soluciones basadas en metaheurísticas poblacionales, como los algoritmos evolutivos o la búsqueda dispersa, reduciendo el tiempo de ejecución sin afectar a la calidad de la solución.

## 1. Introducción

Muchos problemas reales son dinámicos y cambian estocásticamente en el tiempo [2]. Un problema de optimización dinámica es aquel en el que se define una instancia inicial del problema y una colección de “eventos” a lo largo del tiempo que determinan los cambios sobre el problema [13]. Cuando se produce un cambio, por ejemplo en la función objetivo o en las restricciones, también puede cambiar el óp-

timo [2]. En este caso, la solución actual debe adaptarse a las nuevas circunstancias. Por lo tanto, los métodos de optimización utilizados en problemas dinámicos necesitan de estrategias de adaptación. La pregunta clave es cómo utilizar la información encontrada en *eventos* anteriores para obtener nuevas soluciones de alta calidad en el menor tiempo posible.

Las versiones dinámicas de problemas de optimización han recibido menos atención que sus correspondientes estáticos [2]. Por este motivo, los problemas dinámicos a menudo carecen de funciones objetivo bien definidas, datos de referencia estándares y criterios de comparación de soluciones [13, 12, 5]. Recientemente, se han propuesto enfoques basados en metaheurísticas para resolver problemas dinámicos. La estrategia más simple es restablecer el proceso de optimización cada vez que se produce un cambio. Sin embargo, este enfoque consume mucho tiempo, y resulta de poca utilidad práctica [2]. Por otro lado, inicializar el proceso de optimización estrictamente a partir de las mejores soluciones encontradas es peligroso, puesto que se parte de un conjunto con poca diversidad.

El objetivo fundamental del filtrado de partículas es seguir una variable de interés que evoluciona en el tiempo, siguiendo una distribución típicamente no gaussiana y multimodal. La base del método es construir una representación de la función de densidad de probabilidad (pdf) completa [15]. Se ejecuta un conjunto de acciones para modificar el estado de las variables de interés siguiendo un determinado modelo. Además, cada cierto tiempo se dispone de observaciones que restringen el estado de las va-

riables de interés. En el contexto de los filtros de partículas se utilizan múltiples copias de esta variable de interés, cada una con un peso asociado que expresa la calidad de esa muestra en concreto. El filtro de partículas es de naturaleza recursiva, y cada ciclo opera en dos fases: predicción y actualización. Después de cada instante, todas las partículas se modifican utilizando un modelo, que incluye la simulación de ruido de la variables de interés. En la etapa de predicción, los pesos de las partículas se recalculan utilizando la nueva medida.

En este trabajo se propone una hibridación cooperativa de bajo nivel entre los métodos del filtro de partículas y la búsqueda dispersa para su aplicación al TSP dinámico. El algoritmo resultante se ha denominado filtro de partículas con búsqueda dispersa (*Scatter Search Particle Filter - SSPF*). SSPF introduce las estrategias de exploración del espacio de soluciones de la búsqueda dispersa [7] en un algoritmo de filtro de partículas [4, 1]. El algoritmo resultante trabaja en dos etapas bien diferenciadas. La etapa PF propaga y actualiza un conjunto de soluciones a través del tiempo. En la etapa SS, se seleccionan y combinan un subconjunto de estas soluciones, con el objetivo de obtener otras mejores. Los resultados experimentales muestran que el SSPF se puede aplicar con éxito al problema del TSP dinámico.

## 2. El TSP dinámico

El problema del viajante de comercio (*Travelling Salesman Problem - TSP*) consiste en encontrar la ruta más corta que conecta un número fijo de localizaciones o ciudades, visitando exactamente una vez cada una de ellas [6]. El TSP se puede describir mediante un grafo  $G = \{V, E, W\}$ , donde  $V$  es un conjunto de vértices (ciudades),  $E$  es un conjunto de arcos (camino entre ciudades) y  $W$  es una matriz de pesos ( $w_{ij} \in W$  representa el coste del viaje entre las ciudades  $i, j \in V$ ). Formalmente, El TSP se puede enunciar como el problema que consiste en encontrar el circuito hamiltoniano con menor longitud en el grafo  $G$  [17].

El TSP dinámico es una generalización de este problema en el que  $G$  es dependiente del tiempo. Es un problema general y tiene diversas aplicaciones relacionadas con el transporte [6] y el manufacturado [8]. Se han descrito diferentes variedades de

TSP dinámico. La versión considerada en este trabajo mantiene constante el número de ciudades modificando el coste de los grafos entre instancias [6].

Las metaheurísticas más comúnmente aplicadas al TSP dinámico han sido sistemas de hormigas (*Ant Systems - AS*) y computación evolutiva (*Evolutionary Computation - EC*). Existe una gran variedad de implementaciones de AS para el TSP dinámico [13, 6, 8]. AS se sirve de la feromona como mecanismo de transferencia de conocimiento entre instantes. Mediante un proceso de deconstrucción-reconstrucción de la solución, este algoritmo determina qué partes de la solución deben ser descartadas teniendo en cuenta las nuevas restricciones [13]. Muchas implementaciones EC que resuelven el TSP se pueden aplicar al TSP dinámico [17]. EC es muy apropiado para problemas de optimización dinámica porque utiliza conjuntos grandes de soluciones, converge estadísticamente, posee mecanismos de optimización global y una robustez considerable [17]. Existen diferentes estrategias de adaptación de EC a problemas dinámicos, entre los que destacan la adaptación de la mutación, el mantenimiento de memoria implícita o explícita y el mantenimiento de múltiples poblaciones [2].

## 3. Filtro de partículas

El problema de la estimación secuencial desde un punto de vista bayesiano consiste en el cálculo recursivo del estado del sistema  $\mathbf{x}_t$  en el instante  $t$ , utilizando para ello las observaciones  $\mathbf{z}_{0:t} = \{\mathbf{z}_0, \mathbf{z}_1, \dots, \mathbf{z}_t\}$ . Para ello es necesario calcular la pdf  $p(\mathbf{x}_t | \mathbf{z}_{0:t})$ , que se puede obtener en dos etapas:

1. Predicción: Supóngase que se dispone de la pdf  $p(\mathbf{x}_{t-1} | \mathbf{z}_{1:t-1})$  en el instante  $t-1$ . La etapa de predicción implica el uso del modelo del sistema para obtener la pdf a priori  $p(\mathbf{x}_t | \mathbf{z}_{1:t-1})$  en el instante siguiente  $t$ , via la ecuación de Chapman-Kolmogorov:

$$p(\mathbf{x}_t | \mathbf{z}_{0:t-1}) = \int p(\mathbf{x}_t | \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} | \mathbf{z}_{0:t-1}) d\mathbf{x}_{t-1} \quad (1)$$

2. Actualización: En el instante  $t$  se dispone de una nueva medida que puede ser utilizada para actualizar el estado del sistema por medio de la regla de Bayes:

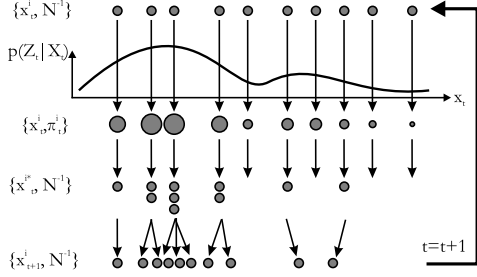


Figura 1: Representación gráfica del funcionamiento de un filtro de partículas.

$$p(\mathbf{x}_t | \mathbf{z}_{1:t}) = \frac{p(\mathbf{z}_t | \mathbf{x}_t) p(\mathbf{x}_t | \mathbf{z}_{1:t-1})}{p(\mathbf{z}_t | \mathbf{z}_{1:t-1})} \quad (2)$$

Las ecuaciones recurrentes (1) y (2) conforman la base de la solución bayesiana óptima. Desgraciadamente, estas ecuaciones no pueden ser determinadas analíticamente, excepto para un conjunto muy restrictivo de casos [1]. Por ejemplo, el filtro de Kalman asume que la pdf a posteriori es siempre gaussiana, y los métodos basados en mallas suponen un número finito y discreto de estados. Sin embargo, el caso general tiene mucho interés en diferentes aplicaciones [4].

Los filtros de partículas (PF) constituyen una clase de métodos de estimación secuencial en los que las distribuciones teóricas de las ecuaciones (1) y (2) se aproximan por medio de muestras simuladas aleatoriamente [4]. Como resultado, se construye una representación discreta de la pdf a posteriori, utilizando un conjunto de muestras ponderadas, llamadas partículas  $\{(x_t^0, \pi_t^0), \dots, (x_t^N, \pi_t^N)\}$ , con pesos normalizados  $\pi_t^n = p(\mathbf{z}_t | \mathbf{x}_t = \mathbf{x}_t^n)$ . En la figura 1 se representa gráficamente el funcionamiento de un filtro de partículas.

Inicialmente, se comienza con una población de partículas aleatorias ponderadas uniformemente  $(\mathbf{x}_t^i, N^{-1})$ , que aproximan la densidad  $p(\mathbf{x}_t | \mathbf{z}_{t-1})$ . En este momento se reciben nuevas medidas  $(\mathbf{z}_t)$  y se calcula el peso para cada partícula, que involucra a la función de densidad de probabilidad  $p(\mathbf{z}_t | \mathbf{x}_t)$ . El resultado es un conjunto de partículas con pesos asociados  $(\mathbf{x}_t^i, \omega_t^i)$  que constituyen una aproximación discreta de  $p(\mathbf{x}_t | \mathbf{z}_t)$ . A continuación, se lle-

va a cabo el paso de remuestreo. En este paso, se seleccionan las partículas con mayor peso para obtener un conjunto de partículas  $(\mathbf{x}_t^{i*}, N^{-1})$  ponderadas uniformemente. El último paso es la predicción, cuyo objetivo es adaptar el conjunto de partículas al nuevo instante, aproximando así la pdf  $p(\mathbf{x}_{t+1} | \mathbf{z}_t)$ .

Por lo tanto, los filtros de partículas son algoritmos que manipulan la evolución de un conjunto de partículas a través del tiempo. Las partículas se mueven de acuerdo a un modelo de movimiento y sobreviven con una probabilidad proporcional a su peso [4].

#### 4. Filtro de partículas con búsqueda dispersa

La optimización dinámica requiere no sólo de estrategias de optimización, sino también de adaptación. Una buena estrategia de adaptación debe tener respuestas acerca de qué información es útil y cómo se transfiere a los siguientes instantes para encontrar buenas soluciones en el menor tiempo posible.

En este sentido, es muy importante tomar una decisión adecuada sobre qué información debe ser transmitida a los siguientes instantes. Una elección incorrecta podría conducir a que el proceso de búsqueda pudiera quedar confinado en las vecindades de un óptimo local, o por otro lado, a despreciar información útil que permitiera encontrar el óptimo más rápidamente. El algoritmo del filtro de partículas con búsqueda dispersa (*Scatter Search Particle Filter - SSPF*) intenta responder a estas preguntas.

La búsqueda dispersa (SS) [7] es un procedimiento metaheurístico basado en formulaciones y estrategias introducidas durante la década de los setenta. El principio de operación de esta metaheurística se basa en que la información sobre la calidad o el atractivo de un conjunto de reglas, restricciones o soluciones se puede utilizar mediante la combinación de éstas en lugar de aisladamente [10]. El motor fundamental de SS consiste en la *combinación de soluciones* de alta calidad, para obtener mejores soluciones que las originales. En este sentido, SS es similar a los métodos evolutivos. Sin embargo, a diferencia de los éstos, SS está fundamentado en la elección sistemática y estratégica sobre un conjunto pequeño de soluciones.

#### 4.1. Hibridación del filtro de partículas y la búsqueda dispersa

SSPF integra mecanismos procedentes de la búsqueda dispersa y los filtros de partículas en dos etapas complementarias:

- La etapa PF está enfocada a la evolución temporal de las mejores soluciones encontradas en instantes previos. Esta etapa evita la pérdida de diversidad en el conjunto de soluciones, permite mantener múltiples hipótesis en un entorno de incertidumbre, y explota el conocimiento de la dinámica del sistema.
- La etapa SS, se dedica a mejorar la calidad de un subconjunto de referencia de buenas soluciones extraído del conjunto de partículas, para explorar de forma eficiente y efectiva el espacio de búsqueda.

La Figura 2 muestra una esquema gráfico del algoritmo SSPF. Las líneas discontinuas engloban las dos componentes principales del esquema SSPF: la evolución temporal PF y la optimización SS. SSPF comienza con una población inicial de  $N$  partículas muestreadas de una función de densidad de probabilidad conocida (Figura 2: etapa de inicialización - INITIALIZE). Cada partícula representa una posible solución al problema de la estimación. Posteriormente, se calculan los pesos de las partículas utilizando una función de ponderación conocida y un vector de medidas obtenidas sobre el sistema (Figura 2: etapa de evaluación - EVALUATE). En este momento se aplica SS, con el objetivo de mejorar las soluciones obtenidas durante la etapa PF. En primer lugar se crea un subconjunto, denominado *RefSet*, formado por las  $b$  ( $b \ll N$ ) mejores partículas obtenidas hasta el momento (Figura 2: etapa de creación del *RefSet* - MAKEREFSET). Entonces se generan y evalúan nuevas soluciones, siguiendo la metodología propuesta por el SS (Figura 2: etapas caminos - PATHS y evaluación - EVALUATE). Posteriormente, se aplica una etapa de mejora sobre las nuevas soluciones obtenidas, para aumentar su calidad (Figura 2: etapa de mejora - IMPROVE). La etapa SS finaliza cuando las nuevas soluciones generadas no mejoran la calidad de aquéllas en el *RefSet*. Cuando esto ocurre, las “peores” soluciones del conjunto de partículas son reemplazadas con las del *RefSet* (Figura 2: etapa de

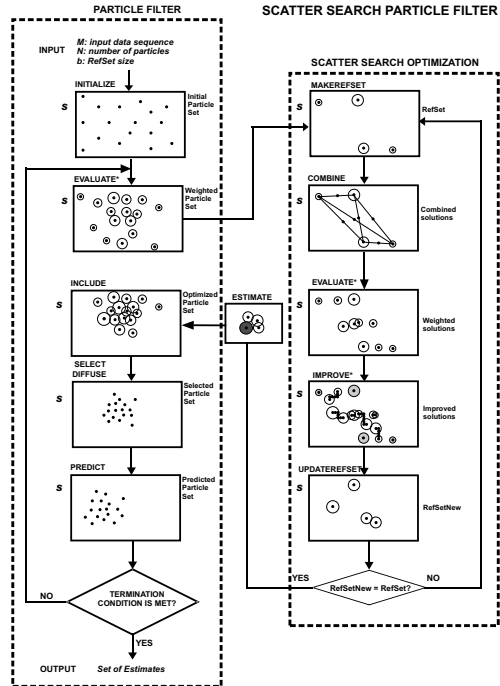


Figura 2: Representación gráfica del funcionamiento de un filtro de partículas con búsqueda dispersa. La evaluación del fitness se lleva a cabo en las etapas EVALUATE e IMPROVE (\*).

actualización - UPDATE). Con este nuevo conjunto de partículas, se ejecutan las diferentes etapas del filtro que conducen a la predicción del conjunto de partículas en el instante siguiente. En primer lugar, se crea un nuevo conjunto como resultado de seleccionar los individuos del conjunto de partículas con una probabilidad proporcional a su peso (Figura 2: etapa de selección - SELECT). Existe la posibilidad de elegir una solución varias veces y por tanto, de pérdida de diversidad en el conjunto de soluciones. Por ello se aplica una etapa de difusión a cada una de las soluciones elegidas (Figura 2: etapa de difusión - DIFFUSE). Finalmente, las partículas son proyectadas en el siguiente instante utilizando la regla de actualización proporcionada por el modelo del sistema (Figura 2: etapa de predicción - PREDICT).

## 4.2. Características del SSPF

La etapa PF es extremadamente útil, no solo cuando se posee conocimiento de la dinámica del sistema. Si se dispone de un modelo de movimiento que describa fielmente la dinámica del sistema, las soluciones pertenecientes al conjunto de partículas van a seguir al óptimo en su evolución temporal. De este modo, la etapa de optimización contará con muy buenas soluciones iniciales y requerirá menos iteraciones para encontrar la solución óptima. En el caso en el que no se disponga de un buen conocimiento de la dinámica del sistema, las soluciones que han sido elegidas para el siguiente instante, se encuentran cercanas a regiones de importancia del espacio de búsqueda. Estas regiones son candidatas a albergar el óptimo en los siguientes instantes, y el filtro de partículas conservará información de cada una de ellas. De esta forma, las soluciones en el conjunto de partículas no estarán tan cercanas al óptimo como las anteriores. A cambio, aseguran que no han descartado ninguna región candidata y mantienen la diversidad necesaria en el conjunto soporte. La etapa de optimización SS comienza con un conjunto de soluciones en *RefSet* que ha conservado su posición en las cercanías de regiones de importancia, por lo que es difícil que el proceso de optimización quede confinado en una región del espacio de búsqueda.

Por lo tanto, SSPF dirige el proceso de optimización a regiones del espacio de soluciones en las que es muy probable encontrar nuevas soluciones que mejoren a las iniciales. PF mantiene una representación discreta de la distribución de la “importancia” sobre el espacio de soluciones. Este conjunto de muestras evita la pérdida de diversidad del conjunto de soluciones y proporciona al SS un conjunto inicial de mucha mejor calidad que otro obtenido aleatoriamente.

SS y PF están mutuamente relacionados de modo que cuando SS mejora, el rendimiento de PF también lo hace, y viceversa. El ajuste de los parámetros en el PF afecta a la relación calidad-diversidad del conjunto de partículas utilizado por SS como conjunto inicial. Por otro lado, SS mejora la calidad del conjunto de partículas, al incluir en él muestras que contribuyen a la mejor estimación de la pdf. Los principales parámetros considerados en el SSPF son los siguientes:

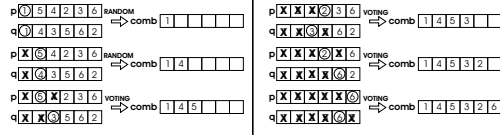


Figura 3: Método de combinación por votos: un ejemplo gráfico.

- El tamaño del conjunto de partículas  $N$ , que debe ser suficiente para asegurar la actualización correcta de la pdf. Su valor se elige en función de la complejidad del espacio de soluciones.
- El tamaño del *RefSet*, típicamente  $b = 10$  [9]. También depende de la dificultad del problema considerado.
- La amplitud de la etapa de difusión  $A$ . El objetivo de esta etapa es evitar la pérdida de diversidad del conjunto de partículas. Habitualmente, consiste en la aplicación de un desplazamiento aleatorio a cada solución, con una desviación  $A$ .

## 5. Implementación del SSPF en el problema del TSP dinámico

El algoritmo SSPF propuesto es genérico se puede aplicar a diferentes problemas de optimización dinámicos. En esta sección se detalla la particularización del SSPF para su aplicación al TSP dinámico.

En nuestra implementación del SSPF cada partícula almacena una posible ruta entre las ciudades. El número de partículas  $N$  en el conjunto de partículas  $S$  se ha elegido teniendo en cuenta el tamaño del problema. Concretamente,  $N$  varía desde  $N = 100$  para instancias de 25 ciudades hasta  $N = 2500$  en instancias de 150 ciudades. En todos los casos, el *RefSet* se construye seleccionando las cinco mejores rutas y las cinco más diversas de  $S$ .

La medida de diversidad se obtiene a través de la distancia para problemas de permutaciones relativos (*R-permutation problems*) utilizada en [9]. El TSP, y por extensión el TSP dinámico, son considerados problemas de permutaciones relativos [9].

Esto significa que la posición relativa de los elementos es más importante que su posición absoluta. La distancia entre dos permutaciones  $p$  y  $q$  para problemas de permutaciones relativos se define como:

$$d(p, q) = \text{num de veces que } p_{i+1} \text{ no sucede a } p_i \text{ en } q \text{ para } i = 1, \dots, n - 1 \quad (3)$$

El método de generación de subconjuntos utilizado en este trabajo devuelve todas las posibles parejas de soluciones que se pueden formar con los elementos del *RefSet*. Como método de combinación de soluciones se ha aplicado el método de votos (*Voting method*) [9] entre parejas de soluciones del *RefSet*. En este método, cada solución de referencia vota por su primer sector no incluido en la solución combinada. El voto determina el elemento que será asignado a la siguiente posición libre de la nueva solución (se puede ver un ejemplo en la figura 3).

Se ha utilizado 2-opt (*2-opt method*) [11, 16] como método de mejora de soluciones. Dada una solución, se consideran todas las parejas de arcos que conectan cuatro ciudades diferentes. Eliminando dos arcos de la ruta, sólo existe una forma de reconstruir las dos subrutas, de modo que la resultante sea un circuito cerrado. La nueva ruta reemplaza a la de partida si es más corta y el proceso se repite hasta que no se produzcan mejoras.

Finalmente, se propone una *actualización estática del RefSet* como método de actualización del conjunto de referencia. Es decir, las nuevas soluciones generadas se almacenan temporalmente hasta que se han combinado todas, y sólo entonces se decide cuáles deben formar parte del *RefSet* siguiendo un criterio basado en calidad.

## 6. Resultados experimentales

Se ha comparado el rendimiento del algoritmo propuesto con diferentes implementaciones de algoritmos evolutivos y búsqueda dispersa. Los experimentos se han realizado utilizando un ordenador Intel Pentium 4 a 1.7 GHz y 256 MB RAM. Todos los algoritmos han sido codificados en MATLAB 7. Estas implementaciones diferentes se han probado y comparado sobre diferentes instancias de TSP dinámico. Las secciones siguientes describen los da-

tos utilizados, los algoritmos y los resultados obtenidos.

### 6.1. Descripción de los datos

Debido a que no existen datos de referencia para el TSP dinámico, se han generado versiones dinámicas a partir de instancias del TSPLIB [14]. En concreto se han utilizado los grafos simétricos euclídeos BAYG29, BERLIN51 y ST70. Cada secuencia se genera a partir del grafo original, obteniendo un grafo en cada instante a partir del previo, introduciendo una perturbación gaussiana sobre la localización de cada ciudad.

### 6.2. Descripción de los algoritmos

Se ha comparado el SSPF con diferentes versiones de SS y MA. Las implementaciones de estos últimos se describen en esta sección.

Se han desarrollado dos implementaciones de SS, denominadas SS1 y SS2. SS1 restablece el proceso de optimización desde el inicio cada vez que se produce un cambio. Esto significa que trata los problemas como independientes, sin tener en cuenta la información obtenida en instancias anteriores. SS2 utiliza el *RefSet* obtenido en instantes anteriores como el conjunto de referencia inicial actual. Esta implementación es totalmente opuesta a la anterior, ya que aquí se supone que la totalidad de la información obtenida en instancias anteriores es aplicable a la actual.

Para ambas implementaciones se ha elegido  $PopSize = 100$  y  $b = 10$ , como se recomienda en [3]. Para poder comparar, utilizamos la misma composición del *RefSet* y los mismos métodos de combinación y mejora en SS1, SS2 y SSPF.

Además, se ha desarrollado una implementación de algoritmo memético, denominado MA, que incluye un mecanismo de mejora. El algoritmo utiliza el método de votos como mecanismo de cruce y 2-opt como método de mejora. Se ha elegido un valor de 100 para  $PopSize$ , probabilidad de cruce  $p_c = 0,25$  y de mutación  $p_m = 0,01$ , tal y como recomienda [11]. Finalmente, la probabilidad de mejora elegida es  $p_i = 0,25$ .

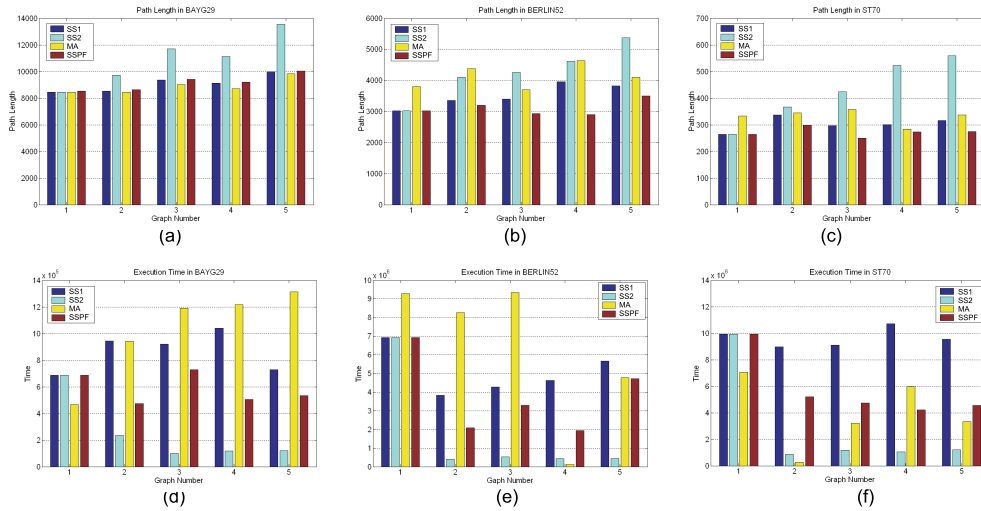


Figura 4: Longitud de las trayectorias en (a) *BAYG29*, (b) *BERLIN52* y (c) *ST70* y tiempo de ejecución requerido en (d) *BAYG29*, (e) *BERLIN52* y (f) *ST70* utilizando SS1, SS2, MA y SSPF.

### 6.3. Resultados obtenidos

La figura 4 presenta los resultados obtenidos para los algoritmos SS1, SS2, MA y SSPF sobre las instancias *BAYG29*, *BERLIN52* y *ST70*. Se debe advertir que la inicialización en SS1, SS2 y SSPF es la misma, por lo que los resultados obtenidos en la primera iteración son exactamente las mismas. Sin embargo, el enfoque MA es diferente, y no encuentra la misma solución en el primer grafo.

La calidad de la solución encontrada por SS1 y SSPF es similar en todas las instancias. Sin embargo, el tiempo de ejecución es significativamente menor en el enfoque SSPF. Los resultados obtenidos por SS2 son de menor calidad, debido a que el proceso de búsqueda queda atrapado en un óptimo local, probablemente en la vecindad del óptimo encontrado previamente. Por esta razón SS2 es el más rápido, pero a costa de conseguir soluciones de muy poca calidad. Finalmente, MA encuentra soluciones de buena calidad, pero el tiempo de cómputo necesario es mayor que el requerido por el SSPF.

En la tabla 1 se resumen los resultados principales obtenidos utilizando los diferentes algoritmos implementados. SSPF presenta la mejor relación

entre el tiempo medio de ejecución y la longitud de las trayectorias.

## 7. Conclusiones

En este trabajo se ha presentado el algoritmo de filtro de partículas con búsqueda dispersa (SSPF). Este método es una hibridación de la metaheurística de la búsqueda dispersa y del filtro de partículas. Está orientado a resolver problemas de optimización dinámica.

La propuesta basa su éxito en la combinación de métodos especializados en problemas de optimización y dinámicos. El filtro de partículas permite modelar la dinámica del sistema y mantener múltiples propuestas en un entorno de incertidumbre, con un coste asumible. La etapa de optimización con búsqueda dispersa refina estas predicciones con el objetivo de encontrar una solución de alta calidad.

Se ha aplicado este algoritmo a una versión dinámica del problema del viajante de comercio (DTSP). Los resultados experimentales han mostrado que el SSPF aumenta de modo apreciable el rendimiento de otras implementaciones basadas en búsqueda dispersa y algoritmos evolutivos en el

Cuadro 1: Tiempo de ejecución promedio y longitud de los circuitos encontradas por SS1, SS2, MA y SSPF en las diferentes instancias

	SS1		SS2		MA		SSPF	
	Longitud	Tiempo	Longitud	Tiempo	Longitud	Tiempo	Longitud	Tiempo
BAYG29	0,86E6	0,91E4	0,25E6	1,09E4	1,03E6	0,89E4	0,58E6	0,91E4
BERLIN52	5,07E6	3,51E3	1,75E6	4,27E3	6,37E6	4,12E3	3,79E6	3,11E6
ST70	302,97	9,65E6	427,58	2,84E6	331,15	3,97E6	272,15	5,72E6

contexto de los problemas de optimización dinámicos. Esta mejora es más significativa cuando aumenta el tamaño del problema considerado.

## Referencias

- [1] Arulampalam, M., Maskell, S., Gordon, S. and Clapp, T., *A Tutorial on Particle Filter for Online Nonlinear/Non-Gaussian Bayesian Tracking*. IEEE Trans. On Signal Processing, 50 (2) (2002) 174-188.
- [2] Branke, J., *Evolutionary Optimization in Dynamic Environments*. Kluwer Academic Publisher (2002)
- [3] Campos, V., Laguna, M., Marti, R., *Scatter Search for the Linear Ordering Problem*. New Ideas in Optimization, McGraw-Hill, 1999.
- [4] Carpenter, J., Clifford, P., Fearnhead, P., *Building robust simulation based filters for evolving data sets*. Technical Report, Dept. Statist., Univ. Oxford, Oxford, U.K, 1999.
- [5] Dror, M., Powell, W., *Stochastic and Dynamic Models in Transportation*. Operations Research, 41 (1993) 11-14.
- [6] Eyckelhof, C., Snoek, M., *Ant Systems for A Dynamic TSP: Ants Caught in a Traffic Jam*. Proc. of ANTS02 Conference.
- [7] Glover, F., *A Template for Scatter Search and Path Relinking*. Lecture Notes in Computer Science, 1363 (1997) 1-53.
- [8] Guntsch, M., Middendorf, M., *Applying Population based ACO to Dynamic Optimization Problems In Ant Algorithms*. Proceedings of Third International Workshop ANTS 2002, LNCS 2463 (2002) 111-122.
- [9] Laguna, M., Marti, R., *Scatter Search: methodology and implementations in C*. Kluwer Academic Publisher, 2003.
- [10] Martí, R., *Procedimientos Metaheurísticos en Optimización Combinatoria*. Matemáticas Vol 1 (1) 3-62 (2000)
- [11] Michalewicz, Z. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1996.
- [12] Sadeh, N., Kott, A., *Models and Techniques for Dynamic Demand-Responsive Transportation Planning*. Technical Report, CMURI-TR-96-09, Robotics Institute, Carnegie Mellon University, 1996.
- [13] Randall, M., *Constructive Meta-heuristics for Dynamic Optimization Problems*. Technical Report. School of Information Technology. Bond University, 2002.
- [14] Reinelt, G., *TSPLIB*. University of Heidelberg. Available online at <http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/>, 1996.
- [15] Rekleitis, I., *A Particle Filter Tutorial for Mobile Robot Localization*. Technical Report TR CIM-04-02. McGill University.
- [16] Vizeacoumar, F., *TSP Implementation*. Project report Combinatorial Optimization CMPUT - 670, 2003.
- [17] Zhang-Can H., Xiao-Lin H., Si-Duo, C., *Dynamic traveling salesman problem based on evolutionary computation*. Proceedings of the 2001 Congress on Evolutionary Computation, 2 (2001) 1283 - 1288.