Un Sistema Híbrido de Metaheurísticas en Entornos Dinámicos

Jose M. Cadenas M^a Carmen Garrido Enrique Muñoz Bernardino Romera

Resumen— **Los problemas dinámicos de optimización están siendo, cada vez más, el centro de la investigación debido a varias causas: muchas de las técnicas existentes no son aplicables, proporcionan nuevos desafios proporcionando un nuevo campo de exploración y además estos problemas suelen ser más semejantes a problemas reales que los problemas estáticos. En este trabajo proponemos un sistema cooperativo definido a partir de un proceso de aprendizaje y compuesto de un conjunto de metaheurísticas, el cual, por su estructura y configuración, se adapta adecuadamente a estos tipos de problemas. Dicho sistema se aplica al problema de la Mochila 0-1 Dinámica.**

Palabras clave— **Sistemas Metaheurísticos Cooperativos, Extracción de Conocimiento, Problemas dinámicos**

I. Introducción

Durante largo tiempo los problemas de Optimización han sido centro de atención de los investigadores, sin embargo este interes ha estado centrado principalmente en los problemas estáticos, es decir, problemas donde el espacio de búsqueda permanece invariable durante todo el tiempo de la búsqueda. No obstante, los problemas reales de optimización son raramente estáticos ya que suelen cambiar con el tiempo, por ejemplo, al buscar la mejor ruta entre dos puntos se ha de tener en cuenta las condiciones del trafico, las cuales pueden variar con el tiempo. Otro ejemplo es el control de un invernadero, [**?**], donde las condiciones medio ambientales cambian con el tiempo. Por esta razón, los problemas dinámicos de optimización (PDOs) mantienen un interés creciente.

Existen distintos algoritmos que se han propuesto para resolver los PDOs, siendo las estrategias inspiradas en la naturaleza las que han tenido más éxito. De hecho, los Algoritmos Evolutivos (AEs) son los más utilizados, [**?**]. Sin embargo, su principal problema es la convergencia de toda la población a una única solución. Esto provoca que los AE tengan dificultades para rastrear la mejor solución con los cambios del problema. Por eso, muchos trabajos tratan de enfrentarse con este problema incrementando la diversidad entre la población usando varias técnicas, [**?**], [**?**], [**?**]. No obstante, los AEs no son los únicos enfoques ya que también podemos encontrar Colonias de Hormigas, [**?**], o Algoritmos de Optimización por Enjambres, [**?**], entre otras metaheuristicas. Sin embargo, otro paradigma interesante, como es el campo de las estrategias cooperativas, no ha sido estudiado a fondo para esta clase de problemas. Solo algunos trabajos, como el aparecido en [**?**], utilizan este enfoque.

Departamento de Ingeniería de la Información y las Comunicaciones. Facultad de Informática. Universidad de Murcia. Spain. E-mail: jcadenas@um.es, carmengarrido@um.es, enriquemuba@dif.um.es, brp89893@alu.um.es

Las estrategias cooperativas que utilizan metaheuristicas se han aplicado a problemas estáticos obteniendo resultados realmente buenos, [**?**], [**?**], mostrando mejoras en robustez y calidad de las soluciones. En este trabajo utilizamos una estrategia cooperativa basada en un sistema de multiagente, donde se utilizan tres metaheurísticas sencillas y diferentes (un Algoritmo Genético, una Búsqueda Tabú y un Temple Simulado), controladoas por un coordinador para resolver un PDO.

El uso de diferentes metaheuristicas permitirá al sistema rastrear mejor el cambio de óptimo por dos principales razones: primero, utilizar varias metaheuristicas incrementa la diversidad de las soluciones, y segundo, el uso de diferentes metaheurísticas con difentes estrategias de búsqueda favorece que al menos una de ellas pueda rastrear el óptimo y guiar a las otras. El modelo del sistema cooperativo que utilizaremos para los PDOs ha sido definido y diseñado para resolver problemas de optimización estáticos, [**?**], en este trabajo queremos mostrar que realizando el sistema más flexible es posible aplicarlo a los PDOs. Por ello, el resto del trabajo se divide de siguiente modo: en la sección **??**, se describe el modelo del sistema Cooperativo, en la sección **??** llevamos a cabo unas pruebas para comprobar la eficiencia de la estrategia y, por último, en la sección **??** exponemos las conclusiones y los trabajos que queremos desarrollar.

II. Una estrategia cooperativa de metaheurístcas basada en aprendizaje

A. La estrategia

En esta sección describiremos brevemente la estrategia o sistema cooperativo, [**?**].

El sistema cooperativo está basado en un sistema multi-agente en el que cada agente implementa una metaheurística que intenta resolver el problema mientras se coordina con los otros agentes. Además, el sistema tiene un agente coordinador que se encarga de controlar y modificar el comportamiento de las metaheurísticas.

El agente coordinador tiene dos tareas fundamentales: recoger la información sobre el rendimiento de

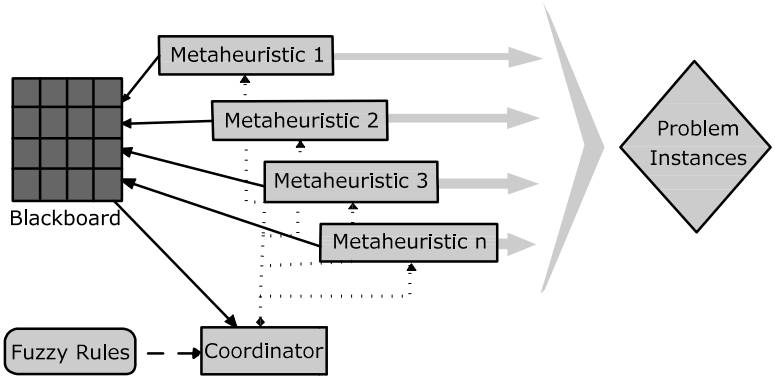

Fig. 1. Un sistema multi-agente de metaheurísticas

cada uno de los agentes resolvedores y enviarles ordenes que afecten a su comportamiento de búsqueda, [**?**], [**?**]. La idea puede verse en la figura **??**.

La comunicación entre los agentes se realiza mediante una arquitectura pizarra, donde cada uno de ellos, tiene asignado un espacio en el cual coloca información periódicamente sobre la solución que ha encontrado del problema. El agente coordinador observa esta información y dirige la búsqueda de cada agente resolvedor.

Para proporcionar inteligencia al agente coordinador utilizamos un proceso de extracción de conocimiento supervisado y esto nos permite definir un conjunto de reglas que modelen el comportamiento de las metaheurísticas cuando resuelven un problema, [**?**]. Una vez aplicado este proceso, se obtiene un sistema completamente operativo para resolver el problema, sin necesidad de aplicar de nuevo el proceso de extracción de conocimiento.

El sistema contendrá, para cada metaheurística, la siguiente regla:

SI $[(wm_1 * d_1$ O ... O $wm_n * d_n)$ ES *suficiente*] ENTONCES cambiar la solución actual de MH

donde:

- n es el número de metaheurísticas.
- MH es la metaheurística que está siendo evaluada por la regla.
- $d_i = \dfrac{(perf_i - perf_{MH})}{maximo(perf_i, perf_{MH})}$ donde

  $perf$ es una medida del rendimiento previamente definida.
- $wm_i \in [0,1]$ donde

  $\sum_{i=1}^{n} wm_i = 1$ y wm_i representa el peso de la meta-heurística i (importancia de la meta-heurística i para resolver la instancia actual).

- *suficiente* es un conjunto fuzzy con función de pertenencia trapezoidal con soporte contenido en $[0,1]$ y definida por la cuadrupla (a,b,c,d):

$$\mu(x,a,b,c,d) = \begin{cases} 0 & x \leq a \ o \ x \geq d \\ \dfrac{x-a}{b-a} & x \in (a,b] \\ \dfrac{d-x}{d-c} & x \in [c,d) \\ 1 & x \in [b,c] \end{cases} \quad (1)$$

Esta regla cambia la posición en el espacio de búsqueda de una meta-heurística que está mostrando un mal rendimiento. Este cambio se realiza hacia una posición cercana a la posición de otra meta-heurística con mejor comportamiento. Como podemos observar en la definición de las reglas, los pesos ponderan la medida de rendimiento. Aunque una metaheuristica se comporte bien puede que no haga disparar la regla debido al bajo peso que tiene con respecto a las otras metaheuristicas.

Además, también se dispone de un conjunto de reglas de inicialización que, dependiendo de la instancia que se intenta resolver, elige los mejores valores de los parámetros para cada meta-heurística.

B. La estrategia adaptable a las instancias

Para obtener los pesos, $wm_i \ \ \forall i$, que definen las reglas y los valores de los parámetros de cada meta-heurística, nosotros utilizamos árboles de decisión fuzzy. Estos árboles reciben una descripción de la instancia que ha de resolverse y como salida nos proporciona el wm_i y la combinación de valores de los parámetros para cada meta-heurística. La idea de como los árboles nos proporcionan la configuración de los distintos conjuntos de las reglas puede verse en la figura **??**.

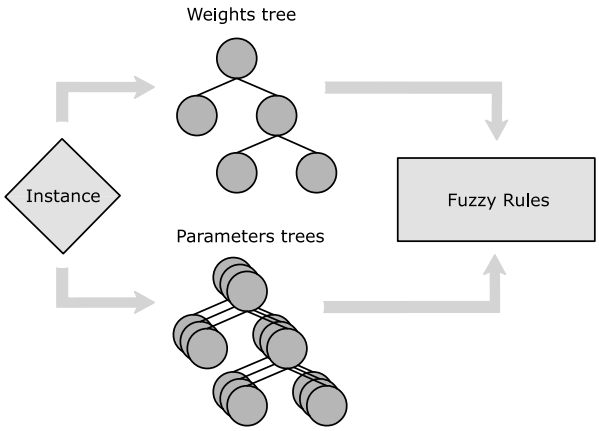

Fig. 2. Obteniendo los pesos y valores de los parámetros

Para disponer de estos modelos de árboles, utilizamos un proceso de extracción de conocimiento, figura **??**, dividido en dos fases: preparación de los datos y minería de datos. En la fase de preparación de los datos buscamos una base de datos que nos propocione información útil y que podamos aplicar las técnicas de Minería de Datos y así obtener un modelo. Esta información se obtiene mediante la resolución de un conjunto de instancias de entrenamiento usando cada meta-heurística con diferentes combinaciones de valores de parámetros. De estas ejecuciones extraemos cierta información a partir de la cual y mediante la fase de minería de datos, obtenemos los árboles descritos anteriormente.

Este tipo de estrategia nos proporciona bastante flexibilidad cuando la instancia cambia o se modifica conforme las metaheuristicas se están ejecutando. En un momento dado, una metaheuristica puede estar guiando la búsqueda de la solución apoyandose las demás sobre ella, y cuando la instancia se modifica, puede cambiar la metaheurística que guía.

C. Obteniendo el sistema cooperativo

El sistema que usaremos para tratar con los PDOs estará compuesto de tres metaheurísticas: un Algoritmo Genético, una Búsqueda Tabú y un Temple Simulado. Con este tipo de sistema estamos aprovechando la bondad de las metaheuristicas basadas en poblaciones y las metaheuristicas basadas en trayectorias.

El sistema está implementado de forma síncrona, es decir, todas las comunicaciones se llevan a cabo en un momento dado, previamente especificado. En ese momento, cada metaheurística escribe su solución actual y el coordinador decide, en base a estas informaciones, que acción debe realizarse. Es importante resaltar la diferencia entre la fase de aprendizaje y la fase de ejecución. En la fase de aprendizaje nostros obtenemos el modelo del agente coordinador. En la fase de ejecución nosotros aplicamos el sistema, previamente modelado, para resolver las instancias del problema, sin necesidad de aplicar de nuevo la extracción del conocimiento.

Al sistema coordinado aprendido lo denominamos KEPS.

Tanto las metaheurísticas como KEPS han sido programadas en Java. Para obtener el motor de inferencia fuzzy que modela al coordinador hemos utilizado la herramienta XFuzzy 3.0, [**?**]. Con esta herramienta modelamos las diferentes reglas y obtenemos un motor fuzzy que fue modificado ligeramente para obtener el motor de inferencia apropiado para nuestro propósito.

Adicionalmente, definimos el conjunto fuzzy "suficiente" que pertenece a la regla previamente definida en la sección **??**:

$suficiente$	conjunto fuzzy con función de pertenencia trapezoidal donde $(a, b, c, d) = (0{,}005, 0{,}01, 1, 1)$.

III. Experimentos

En esta sección, vamos a probar la eficiencia del sistema cooperativo para resolver el problema de la mochila 0-1 dinámico. Este problema ha sido bastante estudiado en la versión estática, y es uno de los problemas dinámicos más típico.

A. El problema de la Mochila 0-1 Dinámico

El Problema de la Mochila 0-1 (PM01) es un problema NP-completo y uno de los problemas de optimización dinámicos clásicos. Por ello, lo hemos escogido como banco de pruebas de nuestra propuesta.

Dicho problema puede definirse formalmente de la siguiente manera: Dado un conjunto de items, cada con un costo y un beneficio, hay que determinar un subconjunto tal que el coste total sea menor que un límite y el beneficio total sea tan grande como sea posible.

La formulación matemática es la siguientes:

$$\begin{aligned} Max \quad & \sum_{i=1}^{n} p_i \times x_i \\ s.a \quad & \\ & \sum_{i=1}^{n} w_i \times x_i \ \leq \ C \\ & x_i \in \{0,1\} \ , i = 1, ..., n \end{aligned}$$

donde

- n es el número de items.
- x_i indica si el item i está o no incluido en la mochila.
- p_i es el beneficio asociado al item i.
- $w_i \in [0, \ldots, r]$ es el peso del item i.
- C es la capacidad de la mochila.

Se asume que $w_i < C, \ \forall i$ y $\sum_{i=1}^{n} w_i > C$.

En sus versiones más comunes, descritas en [**?**], el PM01 Dinámico cambia C con el tiempo. Nuestras pruebas serán realizadas con esta versión.

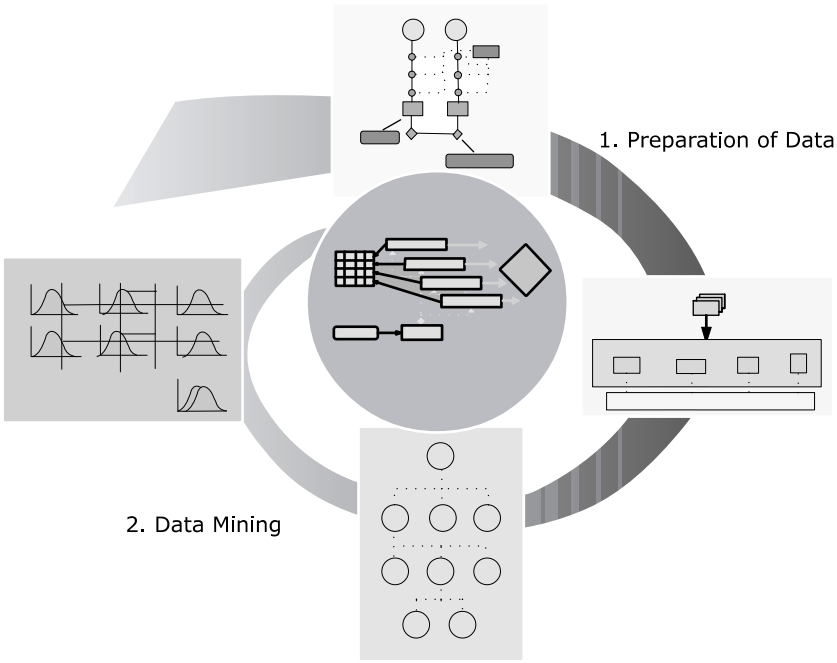

Fig. 3. Proceso de Extracción de Conocimiento

B. Metodología

Para realizar las pruebas, hemos decidido que cada estrategia (metaheurísticas individuales y el sistema cooperativo) se ejecutara durante 200 segundos, modificando la instancia después de 10 segundos. En el caso del sistema cooperativo KEPS, parará cada 250 milisegundos para realizar la coordinación. Cada instancia fue resuelta 10 veces, y los resultados muestran los promedios. El sistema KEPS y las metaheuristicas independiente fueron ejecutados en un Intel core2 Quad 1.66Ghz con 2GB de Memoria.

C. Comparando el sistema KEPS con las metaheurísticas que lo componen

Para realizar las primeras pruebas utilizaremos la instancia mostrada en la tabla **??**, con C cambiando entre 60 y 104 y el óptimo entre 71 y 87. Esta instancia fue definida anteriormente en [**?**] y ha sido utilizada extensamente en la literatura.

En este test comparamos el sistema KEPS con las metaheuristicas individuales que lo componen. Los resultados se muestran en las figuras **??**, **??** y **??**.

La figura **??** muestra una comparación entre el sistema KEPS y el algoritmo Algoritmo Genético (AG), en la que podemos ver como tanto KEPS como el AG encuentran la mejor solución en la situación superior, pero cuando la instancia cambia, el AG no es capaz de mantener el error, mientras que KEPS se mantiene facilmente sobre el cambio de nuevas soluciones.

TABLA I
Instancia del problema de la Mochila 0-1

Número de Objetos	Valor Objeto	Peso Objeto
1	2	12
2	3	5
3	9	20
4	2	1
5	4	5
6	4	3
7	2	10
8	7	6
9	8	8
10	10	7
11	3	4
12	6	12
13	5	3
14	5	3
15	7	20
16	8	1
17	6	2

La figura **??** muestra una comparación entre el sistema KEPS y la búsqueda Tabu (BT), en la que podemos observar como en ambos casos se mantienen sobre la traza de las soluciones cuando cambia el problema. Sin embargo, KEPS es el que obtiene las mejores soluciones. Por último, la figura **??** muestra una comparación entre KEPS y el Temple Simulado (TS). En esta situación el comportamiento es similar al observado con el AG ya que en la situación superior, tanto KEPS como el TS se comportan bien aunque KEPS obtiene mejores soluciones. Pero cuando la solución cambia a la situación inferior el TS no logra mantener los resultados mientras KEPS se mantiene correctamente.

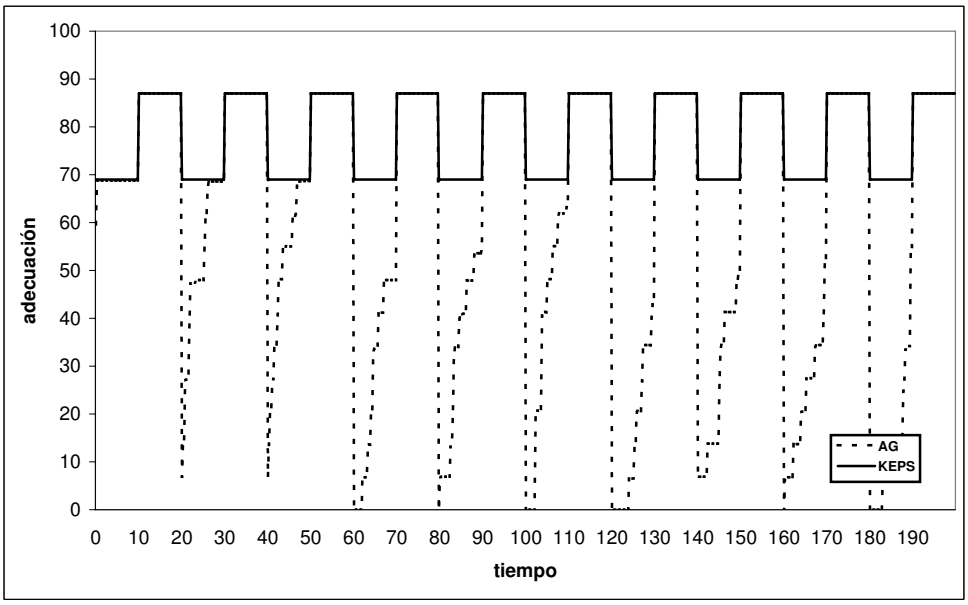

Fig. 4. Comparación entre el Algoritmo Genético y KEPS

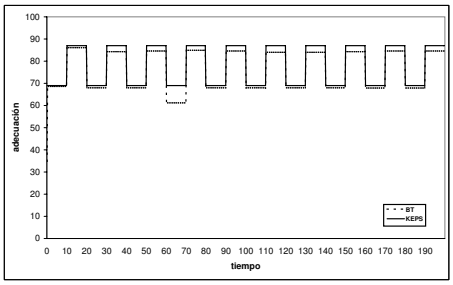

Fig. 5. Comparación entre la Búsqueda Tabú y KEPS

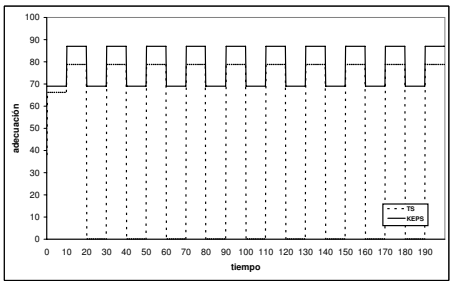

Fig. 6. Comparación entre el Temple Simulado y KEPS

Como hemos podido observar, el sistema KEPS siempre consigue soluciones iguales, por lo menos, a la mejor metaheurística independiente, y eso es importante porque esto es el mínimo demandado del sistema. Pero también podemos observar como en la mayor parte de los casos el sistema KEPS también obtiene una mejora.

Sin embargo, esta instancia parece ser muy sencilla y por eso también probaremos el sistema KEPS con instancias más complejas como las que propone Pisinger [**?**]. Para llevar a cabo estas pruebas nosotros resolvimos una base de datos de instancias compuesta de 20 instancias donde cambiaron tanto el tamaño (número de objetos) como la manera en la que fueron generadas. Con respecto a tamaño, podemos encontrar cuatro tamaños diferentes: 500, 1000, 1500 y 2000 objetos.

En cuanto a la forma en la que han sido generadas existen cinco tipos:

- Spanner: Estas instancias se han construido de forma que sus items son múltiplos de un conjunto pequeño de items llamados key. Esos key fueron generados usando tres distribuciones:
  - Uncorrelated,
  - Weakly correlated,
  - Strongly correlated.
- Profit ceiling: En estas instancias todos los beneficios son multiplos de una parámetro dado d.

▪ Circle: Estas instancias se generan de forma que los beneficios son una función de los pesos teniendo una representación elíptica.

Al igual que con la instancia inicial, hemos cambiado en las distintas ejecuciones el valor de C, oscilando su valor entre el original y el 80 % de este, y obteniendose el óptimo usando el algoritmo exacto Combo [**?**].

Como resultados de estos tests podríamos presentar tantas figuras como instancias, pero debido a las restricciones de espacio nosotros los mostraremos utilizando una medida de rendimiento bastante aceptada para los PDOs: el error off-line [**?**]. Este error es el promedio de las diferencias entre los valores óptimos y las mejores soluciones encontradas en el tiempo:

$$offline\ error = \frac{1}{T}\sum_{t=1}^{T}(optimum - bestSolution)$$

Ejecutado el sistema KEPS y cada una de las metaheurísticas que lo componen sobre las instancias definidas anteriormente, los resultados que obtenemos son los siguientes.

En la tabla **??** mostramos los resultados obtenidos comparando el sistema KEPS y las metaheurísticas. En ella se muestra el error off-line obtenido en la resolución de los tipos diferentes de instancias. En la primera parte de la tabla se muestra el error off-line para cada tipo de instancia, y en el segundo para cada tamaño de la instancia.

Como puede observarse, el sistema KEPS se mantiene muy estable con las distintas ejecuciones de las distintas instancias dinámicas. El sistema coordinado tiene mejor comportamiento que las metaheuristicas individuales, pero por contraste al caso anterior donde las diferencias no eran demasiado grandes, en este caso, como la dificultad de las instancias ha aumentado, las diferencias han llegado a ser más significativas, obteniendo un error entre 1.5 y 3 veces más pequeño que la mejor metaheuristica.

TABLA II
Error off-line del sistema y los metaheurísticas individuales

Tipo	**AG**	**BT**	**TS**	**KEPS**
unc span	0.6213	0.0485	0.5117	0.0021
wea span	0.4774	0.0349	0.5193	0.0216
str span	0.5255	0.1580	0.5266	0.0630
pceil	0.4972	0.0038	0.5024	0.0024
circle	0.5854	0.3545	0.5521	0.1368
Tamaño				
500	0.4784	0.1136	0.5198	0.0371
1000	0.5498	0.1245	0.5278	0.0419
1500	0.5620	0.1122	0.5210	0.0496
2000	0.5783	0.1244	0.5204	0.0530

También hay que notar que el AG y el TS obtienen casi siempre un error por encima del 50 % porque sólo puede rastrear un óptimo, y que este hecho no afecta al comportamiento del sistema KEPS lo que parece indicar que él está cogiendo lo mejor de cada una de las metaheurísticas y lo está combinando de manera adecuada.

KEPS se mantiene muy robusto con las distintas instancias de la mochila 0-1 dinámica.

D. Comparando el sistema KEPS con otros sistemas paralelos

En esta sección queremos comparar el sistema KEPS con otros sistemas.

Uno de ellos es un sistema no cooperativo (no-Coop) en el que se ejecutan de forma paralela las tres metaheurísticas y seleccionamos la mejor solución obtenida por ellas, debemos notar que este resultado no es el mismo que tomar el mejor valor de la tabla **??**, puesto que es posible que en algún caso una metaheurística que de media no es la mejor obtenga el mejor resultado en una ejecución en particular, por lo que los resultados pueden ser mejores. El otro es un sistema cooperativo (CS-WB) en el cual en cada coordinación se cambia la solución de la metaheurística que tiene, en ese momento, el peor comportamiento por la solución de la metaheurística que en ese momento está obteniendo el mejor comportamiento, y que además no hace ningún uso de información aprendida.

Para realizar esta comparación nos centramos en las instancias más complejas, omitiendo la comparación con la primera instancia debido a que por su sencillez los resultados son muy similares.

Como se realizó anteriormente, mostraremos una tabla comparativa (ver tabla **??**) que nos muestra el error promedio de las diferencias entre los valores óptimos y las mejores soluciones encontradas en el tiempo.

TABLA III
Error off-line de KEPS y de otros sistemas

Tipo	**no-Coop**	**CS-WB**	**KEPS**
unc span	0.0461	0.0098	0.0021
wea span	0.0332	0.0332	0.0216
str span	0.1113	0.0974	0.0630
pceil	0.0020	0.0048	0.0024
circle	0.2295	0.1820	0.1368
Tamaño			
500	0.0759	0.0545	0.0371
1000	0.0914	0.0616	0.0419
1500	0.0798	0.0715	0.0496
2000	0.0879	0.0762	0.0530

Como podemos observar, entre los distintos sistemas el que muestra la mejor conducta es el que está controlado por las reglas fuzzy obtenidas por extracción de conocimiento, KEPS. Las diferencias de error entre los dos sistemas cooperativos, CS-WB y KEPS, son estadísticamente significativas según el test no paramétrico de rangos de signos de Wilcoxon a nivel de confianza del 95 %. El error obtenido por el sistema KEPS es menor que el error de CS-WB.

Como puede observarse, KEPS mantiene un comportamiento muy razonable en las instancias más duras (en tipo y tamaño).

Además, las diferencias de error entre las metaheurísticas y KEPS y entre los distintos sistemas paralelos y KEPS son significativas de acuerdo al mismo test no paramétrico de rangos de signos de Wilcoxon a nivel de confianza del 95 %.

Por tanto, hemos probado la utilidad de incorporar información útil para configurar el sistema y el sistema KEPS muestra tener un comportamiento estable y un comportamiento satisfactorio.

IV. Conclusiones y trabajos futuros

En este trabajo hemos propuesto el uso de un sistema cooperativo de metaheuristicas, denominado KEPS, para resolver problemas Dinámicos de Optimización. Debido a la naturaleza de estos problemas, las estrategias de este tipo demuestran ser sistemas adaptables a los cambios y obtienen soluciones muy satisfactorias.

El sistema cooperatico KEPS está compuesto de un Algoritmo Genético, una Búsqueda Tabú y un Temple Simulado cuya búsqueda esta dirigida por un Coordinador. Para modelar al coordinador hemos utilizado un proceso de extracción de conocimiento cuyo resultado ha servido para definir un conjunto de reglas fuzzy que deben controlar el cambio de soluciones y que valores de parámetros tienen que ser utilizados por cada metaheuristica. El uso de metaheuristicas diferentes permite al sistema KEPS rastrear mejor el cambio de solución óptima por dos razones principales: utilizar varias metaheuristicas incrementa la diversidad de las soluciones, y que por lo menos una de ellas puede rastrear el óptimo e informar a las otras.

Además, hay que notar que el sistema KEPS se comporta de manera estable debido, entre otras cosas, a la definición de los árboles de pesos que produce que el sistema se adapte de manera más adecuada a las distintas instancias a las que tiene que enfrentarse.

Los resultados experimentales muestran claramente que el sistema propuesto KEPS es robusto y efectivo, donde siempre obtuvo por lo menos la misma solución que las estrategias individuales que lo componen, y en la mayoría de los casos mejor.

Con respecto a los futuros trabajos tenemos dos líneas a seguir. Una de ellas es la de probar este sistema KEPS con PDOs más complejos como por ejemplo el problema de picos móviles, [**?**]. La otra línea a seguir es la siguiente: Para problemas complejos el costo del proceso de extracción de conocimiento puede ser grande. Por eso proponemos un enfoque diferente en donde podemos aplicar un proceso de aprendizaje por refuerzo. Así tendremos un sistema inicial con un "mal" comportamiento que estará mejorando cuando resuelve más instancias. Este sistema, con aprendizaje por refuerzo, también tiene la ventaja de obtener la información directamente de la ejecución, lo que pensamos puede mejorar el comportamiento del sistema.

Agradecimientos

Los autores agradecen al "Ministerio de Educación y Ciencia" y al "Fondo Europeo de Desarrollo Regional" (FEDER) por el soporte, bajo los proyectos TIN2005-08404-C04-02 y TIN2005-08404-C04-04, para el desarrollo de este trabajo. Así como a la Fundación Séneca, Agencia de Ciencia y Tecnología de la Región de Murcia, que financia a través de una ayuda del Programa Séneca para la formación de personal investigador.

Referencias

[1] E. Alba, G. Luque, F. Luna, Parallel Metaheuristics for Workforce Planning. *Journal of Mathematical Modelling and Algorithms*, 6(3):5009–528, 2007.

[2] D. Angus, T. Hendtlass, Dynamic Ant Colony Optimisation. *Applied Intelligence*, 23:33–38, 2005.

[3] J. Branke, *Evolutionary Optimization in Dynamic Environments*, Kluwer Academic Publishers, 2002.

[4] J.M. Cadenas, M.C. Garrido, L.D. Hernández, E. Muñoz, Towards a definition of a Data Mining process based on Fuzzy Sets for Cooperative Metaheuristic systems. *Proceedings of Conference Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'06)*, 2828–2835, 2006.

[5] J.M. Cadenas, M.C. Garrido, E. Muñoz, A Hybrid System of Nature Inspired Metaheuristics. *Proceedings of Workshop Nature Inspired Cooperative Strategies for Optimization (NICSO 2007)*, 95–104, 2007.

[6] J.M. Cadenas, M.C. Garrido, E. Muñoz, A cooperative Systems of Metaheuristics *Proceedings of 7th International Conference on Hybrid Intelligent Systems (HIS 2007)*, 120-125, 2007.

[7] C. Cruz, D. Pelta, J.L. Verdegay. A fuzzy-set based strategy in dynamic environments. *Proceedings of Conference Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'08)*, 1239–1245, 2008.

[8] D. Dasgupta, D. McGregor, Non Stationary Function Optimization using the Structured Genetic Algorithm. *Proceedings of Parallel Problem Solving From Nature (PPNS-2) Conference*, 145–154, 1992.

[9] W. Du, B. Li B, Multi-strategy ensemble particle swarm optimization for dynamic optimization. *Information Sciences*, 178:3096–3109, 2008.

[10] D.E. Goldberg, R.E. Smith, Nonstationary function optimization using genetic algorithms with dominance and diploidy. *Proceedings of the Second International Conference on Genetic Algorithms*, 59–68, 1987.

[11] A. Le Bouthillier, T.G. Crainic, A cooperative parallel meta-heuristic for the vehicle routing problem with time windows. *Computers and Operations Research*, 32(7):1685–1708, 2003.

[12] F.J. Moreno-Velo, I. Baturone, S. Sánchez-Solano, A. Barriga, XFUZZY 3.0: A Development Environment for Fuzzy Systems. *International Conference in Fuzzy Logic and Technology*, 93–96, 2001.

[13] D. Pelta, C. Cruz, A. Sancho-Royo, J.L. Verdegay, Using memory and fuzzy rules in a cooperative multi-thread strategy for optimization. *Information Sciences*, 176(13):1849–1868, 2006.

[14] D. Pisinger, Where are the hard knapsack problems?. *Computer and Operations Research*, 32(9):2271–2284, 2005.

[15] R.K. Ursem, B. Filipic, T. Krink, Exploring the performance of an evolutionary algorithm for greenhouse control. *Journal of computing and information technology*, 10(3):195–201, 2002.

[16] S. Yang, R. Tinós, A Hybrid Immigrants Scheme for Genetic Algorithms in Dynamic Environments. *International Journal of Automation and Computing*, 4(3):243–254, 2007.