Optimización usando Cúmulos de Partículas aplicado a la distribución de la caña de azúcar en Tucumán

Nicolás Majorel Padilla[(1)(2)], Pamela Álvarez[(2)], Pedro Araujo[(1)], Adrián Will[(1)(2)] y Sebastián Rodríguez[(1)]

[(1)]Grupo de Investigación en Tecnologías Informáticas Avanzadas
Universidad Tecnológica Nacional - Facultad Regional Tucumán
Rivadavia 1050, T4001JJD, San Miguel de Tucumán, Tucumán, Argentina
[(2)]Universidad Nacional de Tucumán - Facultad de Ciencias Exactas y Tecnología
Avenida Independencia 1800, T4002BLR, San Miguel de Tucumán, Tucumán, Argentina
{*nicolas.majorel,pedro.araujo,adrian.will,sebastian.rodriguez*}*@gitia.org*, {*alvpamela47*}*@gmail.com*

Abstract

Los algoritmos de optimización basados en Cúmulos de Partículas son muy utilizados en el campo de la optimización, gracias a que su implementación es sencilla y su convergencia al óptimo presenta una buena performance. Este trabajo presenta un algoritmo novedoso que resulta de la combinación entre un Cúmulo de Partículas y Sistemas Multiagentes, utilizado para resolver una variante no lineal del Generalized Assignment Problem cuyo objetivo es optimizar la distribución de caña de azúcar en Tucumán, considerada una de las actividades económicas más destacadas de la provincia.

Los resultados obtenidos de aplicar dicho algoritmo sobre un conjunto de pruebas estándar, muestran que es capaz de encontrar el valor óptimo en todos los casos manteniéndose siempre en tiempos acotados.

Palabras clave: *Particle Swarm Optimization, Sistemas Multiagentes, GAP no lineal.*

1. Introducción

El presente trabajo es una continuación de un trabajo previo presentado por los mismos autores [17]. En dicho trabajo se introdujo una variante no lineal de un problema conocido, el *Generalized Assignment Problem (GAP)*, se presentó un conjunto de instancias de prueba junto con sus resultados óptimos, para que sirva como referencia a futuros algoritmos que se prueben sobre el problema, y se hizo una modelización del problema utilizando un enfoque organizacional de Sistemas Multiagentes.

Considerando esos antecedentes, en este trabajo se expone un algoritmo heurístico como forma de encontrar soluciones al problema planteado. El algoritmo propuesto es un algoritmo de Optimización mediante Cúmulo de Partículas (*Particle Swarm Optimization* o PSO) en su versión Binaria. El algoritmo resulta un híbrido entre un algoritmo PSO Binario y un Sistema Multiagentes (SMA), aplicado a una variante no lineal del *GAP*, y que tiene como objetivo la optimización de la distribución de la caña de azúcar en la provincia de Tucumán, Argentina.

Con ese algoritmo híbrido se realizan las pruebas sobre el conjunto de instancias mencionado anteriormente, y se comparan los resultados obtenidos, verificando que el algoritmo propuesto encuentra las soluciones óptimas, en tiempos razonables.

El trabajo se estructura de la siguiente manera. A continuación, en la sección 2 se realiza una descripción del GAP, de la variante a resolver, y de trabajos sobre PSO basados en Sistemas Multiagentes. En la sección 3 es descripto con detalles el algoritmo de PSO implementado, y en la sección 4 se explica la metodología de pruebas utilizada, y los resultados obtenidos.

2. Trabajos relacionados

Esta sección se divide en tres partes: en la primera de ella se detallan trabajos vinculados al GAP, su definición, sus variantes y algoritmos propuestos para resolverlos; luego se hace un breve repaso de nuestra variante del problema; finalmente, se detallan trabajos sobre PSO basados en Sistemas Multiagentes.

2.1. GAP: definiciones y variantes

Como ya mencionamos anteriormente, el problema es una variante del *Generalized Assignment Problem (GAP)*, que es definido en [6] como "un problema de optimización combinatoria bien conocido, NP-completo, que consiste en encontrar la asignación de tareas a agentes que genere mínimos costos, de manera que cada tarea es asignada a exactamente un agente, sujeto a la capacidad de dicho agente".

Una manera de representar el problema es mediante la notación empleada en [34]: n representa el número de agentes (para nuestro problema, ingenios) y m el número de tareas (parcelas) que pueden ser asignadas a los agentes. Cuando una parcela i es asignada al ingenio j se expresa como x_{ij}, y se genera un costo c_{ij} o un beneficio p_{ij} asociado a dicha asignación. Además, cada ingenio tiene una capacidad máxima b_j, y una parcela i le aporta una cantidad de producción expresada como a_{ij}. El GAP busca encontrar una asignación de todas las parcelas en los ingenios, de manera que la suma de las producciones aportadas por todas las parcelas asignadas a un ingenio no exceda la capacidad máxima del mismo, y el costo total de la asignación sea mínimo, manteniendo siempre que cada parcela debe ser asignada a un único ingenio.

El GAP es un problema NP-hard [9] y el problema de saber si existe una solución válida es *NP-completo* [18]. En la literatura pueden encontrarse muchos algoritmos determinísticos y heurísticos desarrollados [5, 18, 19, 21, 31, 30, 34] y también diversas variantes, entre las que se toman relevancia por estar relacionadas con este trabajo las siguientes: el *Minimum Quantity GAP* [13] donde la capacidad de un ingenio tiene una cota mínima; el *Multi-resource GAP* [16] en el que más de una recurso es asignado a cada ingenio; yel *Additive Non-Linear GAP* [23] en el que se añade un término no lineal a la función, entre otros [4, 8, 15, 22].

2.2. Variante no lineal del GAP: breve repaso

A pesar de todas las variantes del GAP analizadas, ninguna es suficiente para modelar correctamente el problema planteado, porque el mismo posee las siguientes diferencias:

- Cada ingenio posee una capacidad mínima.
- Existen deterioros o pérdidas de materia prima en los ingenios.
- La mayoría de los ingenios en Tucumán alcanzan su rendimiento pico al trabajar entre $60\,\%$ y $95\,\%$ de su capacidad máxima teórica.
- La eficiencia de un ingenio depende de manera no lineal del régimen de trabajo al que se ve sometido.
- El modelo se limita a manejar la distribución de la producción de un solo día.

Por lo tanto, la variante del GAP introducida en [17] quedaría expresada de la siguiente manera:

$$\text{maximizar } Z(X) = \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}x_{ij}h_j \left(\sum_{i=1}^{n} a_{ij}x_{ij}\right) \quad (1)$$

sujeto a:

$$d_j \leq \sum_{i=1}^{n} a_{ij}x_{ij} \leq b_j \quad j = 1,\ldots,m \quad (2)$$

$$\sum_{j=1}^{m} x_{ij} = 1 \quad i = 1,\ldots,n \quad (3)$$

$$x_{ij} = \begin{cases} 1, & \text{si parcela i es asignada a ingenio j} \\ 0, & \text{en otro caso.} \end{cases} \quad (4)$$

Donde:

- $Z(X)$ es la eficiencia global del sistema.
- a_{ij} es la producción en toneladas que la parcela i entrega al ingenio j. Depende del ingenio j para modelar las pérdidas ocasionadas por el ingenio mismo o por las rutas.
- d_j es la capacidad mínima que necesita el ingenio para mantener su producción.
- h_j representa la eficiencia a la cual se encuentra trabajando el ingenio j, en función del porcentaje de su capacidad máxima. La función h_j utilizada fue una curva gaussiana con una media en $0{,}85$ y una desviación estándar de $0{,}25$ definida para el intervalo [0.30, 1.00], y con el valor cero fuera de tal intervalo.

2.3. Uso de PSO en Sistemas Multiagentes

Los Sistemas Multiagentes han demostrado en las últimas décadas su enorme potencial abordando una amplia gama de problemas que va desde la robótica hasta la resolución de problemas distribuidos, entre otros [29]. El agente es el componente principal del paradigma, puede ser considerado como una entidad física o virtual con un alto grado de autonomía, independiente, capaz de cooperar, competir, comunicarse, actuar flexiblemente y ejercer control sobre su comportamiento dentro del marco de sus objetivos. Estas entidades evolucionan dentro de un ambiente al cual son capaces de percibir y modificar. En contraste, un sistema multiagentes está caracterizado por múltiples agentes inteligentes que interactúan entre sí para alcanzar objetivos que pueden ser compartidos o no entre los agentes [28].

Existe en la literatura pocos ejemplos del uso de algoritmos PSO junto a Sistemas Multiagentes. A continuación se describirán brevemente los trabajos que consideramos más significativos sobre el tema.

La distribución de energía reactiva en sistemas de potencia es un problema de optimización combinatorio complejo que involucra restricciones no lineales y discontinuas. El método que propone Zhao [33] integra exitosamente los Sistemas Multiagentes y los algoritmos PSO, a los que denomina MAPSO (Multiagent-based Particle Swarm Optimization), y fue desarrollado para determinar la solución óptima global de alta calidad (o cercana a la óptima) del problema. El algoritmo fue sometido a pruebas usando los casos de pruebas de la IEEE[1], Sistema de Energía 30-bus y 118-bus, cuyos resultados demostraron converger en las mejores soluciones mucho más rápido que los enfoques previos propuestos para este tipo de problema.

Para los problemas de ubicación y recuperación de objetivos en ambientes desconocidos mediante robots automáticos, Pugh [20] presenta un algoritmo multi-robot inspirado en los PSO que permite minimizar el proceso de búsqueda, el cual es virtualmente un Sistema Multiagentes dado que cada robot es autónomo e interactúan entre sí para cubrir todo el espacio de búsqueda. El autor destaca los beneficios de un sistema multi-robot por sobre un mecanismo basado en un único robot: masificación de la búsqueda gracias al paralelismo, decremento significativo de los tiempos de localización de objetivos, mejora de la robustez contra fallas debido a la redundancia, y por último, presenta una buena escalabilidad de rango y robustez por cada nuevo agentes que se agrega.

En el trabajo realizado por Wang en [27], propone un nuevo sistema de control el cual toma en consideración aspectos de la gestión de comfort y la energía aplicados al campo de la domótica. Está basado en un sistema multiagentes jerárquico usado para construir el sistema de control multicapas. En este problema en particular, el PSO es utilizado para optimizar el conjunto de puntos para dicho sistema.

Por último, Kumar en [14] propone un nuevo algoritmo conocido como HMAPSO (Hibryd MultiAgent-based Particle Swarm Optimization) diseñado para resolver el problema económico de la distribución de la energía. Dicho algoritmo es el resultado de la combinación de los SMA, un algoritmo PSO y el proceso de toma de decisión de las abejas. Para demostrar la robustez, velocidad y precisión del algoritmo, los autores, realizan una comparativa con otros PSO similares con exactamente las mismas restricciones quedando demostrado, en todos los casos, la efectividad del HMAPSO.

Al igual que los trabajos anteriores nuestro enfoque representa un nuevo híbrido entre un algoritmo PSO básico, un sistema multiagentes y una heurística variante no lineal del *Generalized Assignment Problem*.

[1] Institute of Electrical and Electronics Engineers

3. Algoritmo propuesto

En esta sección se describirá el algoritmo híbrido planteado. Para ello, primero se explicarán generalidades de PSO definiendo sus componentes y comportamientos, luego se describirá su variante binaria, mostrando un pseudocódigo; y finalmente, se detallará su vinculación con los Sistemas Multiagentes.

3.1. Generalidades de PSO

El algoritmo de Optimización mediante Cúmulo de Partículas (*Particle Swarm Optimization* o PSO) [11], es un algoritmo bioinspirado, que se basa en una metaheurística inspirada en una analogía con el comportamiento social de ciertas especies, como el vuelo de las bandadas de aves o el movimiento de los bancos de peces, y se ha popularizado gracias a su rapidez de convergencia y la simplicidad de su implementación. En los últimos años surgieron diversos trabajos en los que se utilizan PSO en el campo de la optimización tradicional [10], en optimización multiobjetivo [32], y en optimización dinámica [2, 3], entre otros.

PSO es un algoritmo basado en población, pero no es un algoritmo evolutivo, porque carece del proceso de selección característico de estos. En PSO, un determinado agente (o partícula) representa una posible solución al problema a resolver, forma parte de un conjunto de agentes de similares características, y debe tomar decisiones sobre su comportamiento. Estas decisiones se realizan teniendo en cuenta un componente individual de cada agente, que representa los resultados de sus exploraciones anteriores, y un componente social que indica cómo influyen en el agente el resto de los individuos del conjunto. Mediante estos factores, se determina una dirección hacia dónde debe moverse el agente, para alcanzar una nueva posición dentro del espacio de soluciones del problema.

Es importante destacar que en PSO se utiliza una población de tamaño fijo, que el espacio de soluciones se encuentra definido de antemano y que no se permite que los agentes se desplacen fuera del mismo, y que las partículas están constantemente en exploración.

Adicionalmente, también es importante notar que PSO es una metaheurística que hace muy pocas suposiciones con respecto al problema que se busca resolver y es capaz de buscar en espacios de soluciones muy grandes. Pero como contrapartida, debe mencionarse que metaheurísticas como PSO no garantizan que se alcance la solución óptima al problema.

En su aspecto más general, cada partícula está compuesta por los siguientes elementos:

- Un vector X_i que almacena la posición actual de la partícula en el espacio de búsqueda.
- Un vector $pBest_i$ que almacena la mejor solución encontrada por la partícula hasta el momento.
- Un vector de velocidad V_i que almacena la dirección según la cual se moverá la partícula.
- El valor $fitness\_x_i$ que indica cuan buena es la solución actual.
- El valor $fitness\_pBest_i$ que indica el valor de la mejor solución encontrada hasta el momento.

La población se inicializa generando aleatoriamente las posiciones y las velocidades iniciales de las partículas. Luego, siguiendo un proceso iterativo, las partículas comienzan a moverse por el espacio de búsqueda hasta encontrar una nueva posición. Con esa nueva posición, se calcula y actualiza su valor $fitness\_x_i$, y si este valor es mejor que el mejor encontrado hasta el momento, se actualizan los valores de $pBest_i$ y $fitness\_pBest_i$.

En cada iteración, la velocidad de cada partícula se actualiza utilizando la Ecuación 5, en la que se distinguen tres términos: el primero representa la tendencia de un individuo a seguir explorando en la misma dirección (inercia) [24]; el segundo representa la tendencia a permanecer próximo a posiciones que le dieron buen resultado anteriormente; mientras que el tercer término representa la influencia que ejercen el resto de las partículas o la imitación del mejor individuo conocido por toda la población.

$$\begin{aligned} V_i(t+1) = & \; w * V_i(t) \\ & + c_1 * rand_1 * (pBest_i - X_i(t)) \\ & + c_2 * rand_2 * (gBest_i - X_i(t)) \end{aligned} \tag{5}$$

Puesto que los vectores varían en cada iteración, se utiliza la variable t para indicar la iteración a la que se hace referencia. Al valor w se lo denomina como factor de inercia. Las constantes c_1 y c_2 son parámetros fijos del algoritmo, conocidas como constantes de aceleración, y normalmente tienen el mismo valor. Los factores $rand_1$ y $rand_2$ son números aleatorios entre 0 y 1. La variable, $gBest_i$ (el mejor global) se corresponde con el valor de la mejor solución encontrada hasta el momento por las demás partículas.

Cuando la velocidad de todas las partículas se ha actualizado, se suma con la posición actual de la partícula para moverla a su nueva posición en la siguiente iteración, según la Ecuación 6.

$$X_i(t+1) = X_i(t) + V_i(t+1) \tag{6}$$

Existen muchos trabajos relacionados con la elección de los valores w, c_1 y c_2, que son críticos para la convergencia del algoritmo [7] [26].

En [24] se concluye que cuando el valor de w es pequeño, el algoritmo realiza una búsqueda más localizada y depende fuertemente de la inicialización. A medida que crece este valor, las búsquedas son más amplias, pero hay más dificultad en encontrar el óptimo. Así mismo, se propone el uso de un valor de w decreciente con el tiempo. Con posterioridad, esta propuesta se analizó de forma detallada en [25] y en [32], concluyéndose que si w es creciente o decreciente depende fuertemente del problema a resolver.

3.2. PSO Binario

Posteriormente, en [12] los autores proponen una versión binaria del algoritmo, con el objetivo de extender el campo de aplicación a problemas de optimización combinatoria. La modificación propuesta afecta a la codificación de las partículas, y a la fórmula de actualización de la posición. En esta variante, la velocidad es utilizada como entrada de una función sigmoide que indica la probabilidad que la posición tome el valor 1. Luego, se genera un nuevo número aleatorio y la nueva posición de la partícula se determina según la ecuación 7.

Un pseudocódigo del PSO Binario se muestra en el algoritmo 1:

$$X_i(t+1) \;\; = \;\; \begin{cases} 1, & \text{si } rand() < sig(V_i(t+1)) \\ 0, & \text{en otro caso.} \end{cases} \tag{7}$$

3.3. Vinculación con SMA

En nuestro enfoque existe una relación uno a uno entre un agente y una partícula. A su vez, cada uno de estos agentes está identificado con una posible solución del problema de optimización. Son necesarios dos tipos de agentes en nuestro modelo: *Swarm* y *Partícula*. El primero -de única instancia en la simulación- es el responsable de la creación de todas las instancias del agente *Partícula* definidas en la población y que son requeridas por el problema. El número de instancias varía según las restricciones del problema, y es mayor cuando el sistema no realiza una adecuada exploración del espacio del problema. Además, *Swarm* recibe los valores de la posición actual de cada partícula. En cuanto al agente *Partícula*, éstos reciben los valores de inicialización de parte del agente *Swarm*, y una vez obtenido este valor, calculan su vector velocidad V_i, su nueva posición X_i y finalmente su valor de $fitness\_x_i$. Este último valor es devuelto al agente *Swarm* para que determine cuál es el nuevo óptimo del conjunto de partículas instanciadas.

Tabla 1. Resultados del Sistema de Partículas para el problema propuesto

Tipo	m	n	Optimo	Partículas	Iteraciones	Parámetros	Obtenido	t[s]	Confiabilidad
A	3	12	0,658656	100	500	[0.8,1.5];1.43;1.43	0,658656	103,647	100 %
A	4	10	0,497642	200	500	[0.8,1.5];1.43;1.43	0,497642	179,728	100 %
A	5	10	0,544364	100	500	[0.8,1.5];1.43;1.43	0,544364	110,952	100 %
A	3	15	0,871947	100	500	[0.8,1.5];1.43;1.43	0,871947	103,558	100 %
A	4	12	0,628255	200	500	[0.8,1.5];1.43;1.43	0,628255	188,168	100 %
B	3	12	0,543980	500	1000	[1.9,1.2];1,30;1.30	0,543980	908,177	100 %
B	4	10	0,614256	100	500	[1.9,0.7];1.30;1.30	0,614256	99,546	90 %
B	5	10	0,542515	600	1000	[1.9,1.5];1,30;1.30	0,542515	1145,751	80 %
B	3	15	0,675879	100	1000	[1.9,0.7];1.30;1.30	0,675879	195,993	80 %
B	4	12	0,606471	200	500	[1.9,0.7];1,30;1.30	0,606471	178,618	100 %
C	3	12	0,564030	100	1000	[1.9,0.7];1,30;1.30	0,564030	157,779	100 %
C	4	10	0,459722	200	1000	[1.9,0.7];1,30;1.30	0,459722	349,569	90 %
C	5	10	0,473988	200	1500	[1.9,0.7];1,30;1.30	0,473988	499,459	90 %
C	3	15	0,535528	300	1000	[1.9,0.7];1,30;1.30	0,535528	496,029	80 %
C	4	12	0,501473	600	1000	[1.9,1.2];1,30;1.30	0,501473	1049,213	90 %
D	3	12	0,277288	100	1000	[1.9,0.7];1,30;1.30	0,277288	197,566	100 %
D	4	10	0,284239	1000	500	[1.9,1.2];1,30;1.30	0,284239	1318,644	90 %
D	5	10	1,106979	300	1000	[1.9,0.7];1,30;1.30	1,106979	546,871	100 %
D	3	15	0,308865	200	1000	[1.9,0.7];1,30;1.30	0,308865	356,877	80 %
D	4	12	0,284524	300	2000	[1.2,1.9];1,30;1.30	0,284524	1052,803	50 %

Data: Pop
Result: la mejor solución encontrada
Pop = CrearPoblacion(N);
while *no se alcance la condición de terminación* **do**
  for *i = 1 to size(Pop)* **do**
    Evaluar Partícula X_i de Pop;
    if *fitness(X_i) es mejor que fitness($pBest_i$)* **then**
      $pBest_i$ = X_i;
      fitness($pBest_i$) = fitness(x_i);
    end
  end
  for *i = 1 to size(Pop)* **do**
    Elegir $gBest_i$;
    Calcular nuevo V_i;
    for *d = 1 to n* **do**
      if *rand() < sig(V_{id})* **then**
        X_{id} = 1
      end
      else
        X_{id} = 0
      end
    end
  end
end

Algoritmo 1: Pseudocódigo del Algoritmo PSO Binario

4. Metodología y Resultados

En la sección 2.1 se detalló la variante del GAP que se pretende resolver, y las pruebas realizadas se basaron en las instancias disponibles en [1]. Las mismas son idénticas a las que se pueden encontrar en la literatura [6, 31], pero de menor tamaño. Las instancias utilizadas pertenecen a los tipos A, B, C y D; siendo las de tipo D las más difícil de resolver porque c_{ij} y a_{ij} son inversamente proporcionales. En total se cuentan con 20 instancias diferentes de complejidad media que resultan representativas del conjunto total.

Las pruebas se realizaron en una PC con un microprocesador Intel Core i7-5500U @2.4 GHz, con 6 GB de memoria RAM y un sistema operativo de 64 bits. Para cada instancia se realizaron diez ejecuciones del algoritmo de PSO, independientes entre sí, y los resultados obtenidos se resumen en la Tabla 1.

En dicha Tabla, las columnas *Tipo*, *m* y *n*, hacen referencia al tipo de instancia utilizado, su cantidad de ingenios y su cantidad de parcelas, respectivamente. La columna *Óptimo* indica el máximo valor obtenido para cada instancia mediante un algoritmo de fuerza bruta [17]. Luego, las siguientes columnas muestran la cantidad de partículas usadas en el sistema, la cantidad de iteraciones efectuadas, y parámetros específicos del PSO. En particular, en la columna *Parámetros* se pueden identificar los tres valores que identifican al Sistema de Partículas, separados por punto y coma: el rango en el cual varía el parámetro w, y los valores de los paráme-

tros c_1 y c_2, que son siempre iguales.

La siguiente columna de la Tabla, *Obtenido*, muestra el mejor valor obtenido mediante el algoritmo propuesto. Es importante destacar que para todas las instancias, el máximo valor obtenido coincide con el valor óptimo de referencia. Finalmente, las últimas dos columnas indican el tiempo promedio requerido por las ejecuciones y el porcentaje de las ejecuciones en las que el sistema encontró el valor óptimo de referencia. Este valor idealmente debería ser 100 %, pero por las características del algoritmo, es probable que a veces las partículas se queden estancadas en un máximo local y no encuentren el máximo global. Puede observarse que para todas las instancias este valor es igual o superior al 80 %, a excepción de la instancia de Tipo D, con 4 ingenios y 12 parcelas, en la que pudo obtenerse el valor óptimo solamente en el 50 % de las pruebas.

5. Conclusiones

En este trabajo se introdujo una nueva técnica heurística, que es la combinación de un Sistema de Cúmulo de Partículas Binario usando Sistemas Multiagentes, para resolver una variante no lineal del *Generalized Assignment Problem*. Dicha variante tiene como objetivo la optimización de la distribución de la caña de azúcar en la provincia de Tucumán, Argentina.

Se realizaron 200 pruebas con ese algoritmo sobre un conjunto de instancias particular para el problema, y pudo alcanzarse el resultado óptimo el 91 % de las veces. De los resultados obtenidos, se puede concluir que este algoritmo híbrido resuelve el problema planteado, encontrando el valor óptimo para todas las instancias, en la mayoría de las veces, y en tiempos aceptables.

Una posible continuación del trabajo consiste en repetir las pruebas pero utilizando instancias más grandes del problema. Si bien la escalabilidad de los SMA es ampliamente conocida, en este caso la necesidad de particulas del PSO hace que la escalabilidad sea compleja debido al numero de particulas necesaria.

Agradecimientos

Este trabajo fue parcialmente financiado por la Universidad Tecnológica Nacional mediante el proyecto UTN-PID 1318.

Referencias

[1] Benchmarks for vpgap, 2014. `http://www.gitia.org/projects/vpgap/`.

[2] T. M. Blackwell and Peter J. Bentley. Dynamic search with charged swarms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '02, pages 19–26, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[3] Tim Blackwell. Particle swarm optimization in dynamic environments. In Dr Shengxiang Yang, Dr Yew-Soon Ong, and Dr Yaochu Jin, editors, *Evolutionary Computation in Dynamic and Uncertain Environments*, number 51 in Studies in Computational Intelligence, pages 29–49. Springer Berlin Heidelberg, 2007.

[4] J. Geunes C. Rainwater and E. H. Romeijn. The generalized assignment problem with flexible jobs. *Discrete Applied Mathematics*, 157(1):49–67, 2009.

[5] Dirk G Cattrysse and Luk N Van Wassenhove. A survey of algorithms for the generalized assignment problem. *European Journal of Operational Research*, 60(3):260–272, 1992.

[6] Paul C Chu and John E Beasley. A genetic algorithm for the generalised assignment problem. *Computers & Operations Research*, 24(1):17–23, 1997.

[7] M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, 2002.

[8] P. K. De and Bharti Yadav. A general approach for solving assignment problems involving withfuzzy cost coefficients. *Modern Applied Science*, 6 number 3:2–10, 2012.

[9] Marshall L Fisher, Ramchandran Jaikumar, and Luk N Van Wassenhove. A multiplier adjustment method for the generalized assignment problem. *Management Science*, 32(9):1095–1103, 1986.

[10] Xiaohui Hu, Yuhui Shi, and R. Eberhart. Recent advances in particle swarm. In *Congress on Evolutionary Computation, 2004. CEC2004*, volume 1, pages 90–97 Vol.1, 2004.

[11] J. Kennedy and R. Eberhart. Particle swarm optimization. In *, IEEE International Conference on Neural Networks, 1995. Proceedings*, volume 4, pages 1942–1948 vol.4, 1995.

[12] J. Kennedy and R. C. Eberhart. A discrete binary version of the particle swarm algorithm. In *, 1997 IEEE International Conference on Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation*, volume 5, pages 4104–4108 vol.5, 1997.

[13] Sven O Krumke and Clemens Thielen. The generalized assignment problem with minimum quantities. *European Journal of Operational Research*, 228(1):46–55, 2013.

[14] Rajesh Kumar, Devendra Sharma, and Abhinav Sadu. A hybrid multi-agent based particle swarm optimization algorithm for economic power dispatch. *International Journal of Electrical Power & Energy Systems*, 33(1):115–123, 2011.

[15] J.F. Cordeau L. Moccia, M.F. Monaco and M. Sammarra. A column generation heuristic for a dynamic generalized assignment problem. *Computers & Operations Research*, 36(9):2670–2681, 2009.

[16] Manuel Laguna, James P Kelly, JoséLuis González-Velarde, and Fred Glover. Tabu search for the multilevel generalized assignment problem. *European Journal of Operational Research*, 82(1):176–189, 1995.

[17] Nicolás Majorel Padilla, Pedro Araujo, Adrian Will, and Sebastian Rodriguez. Modelización no lineal del problema de la distribución de la caña de azúcar y su implementación en sistemas multiagentes organizacionales. In *3er Congreso Nacional de Ingeniería Informática/Sistemas de Información*, Buenos Aires, Argentina, 2015.

[18] Silvano Martello and Paolo Toth. *Knapsack problems*. Wiley New York, 1990.

[19] Ibrahim H Osman. Heuristics for the generalised assignment problem: simulated annealing and tabu search approaches. *Operations-Research-Spektrum*, 17(4):211–225, 1995.

[20] J. Pugh and A. Martinoli. Inspiring and modeling multi-robot search with particle swarm optimization. In *2007 IEEE Swarm Intelligence Symposium*, pages 332–339, 2007.

[21] H Ramalhinho and Daniel Serra. Adaptive search heuristics for the generalized assignment problem. *Mathware & soft computing*, 9(3):209–234, 2008.

[22] Kamran Shahanaghi, Hamidreza Haddad, Mohammad Hossein Babaie, and Camran Alavi. A new approach for solving the generalized assignment problem in uncertain environment. *Advances in Mathematical and Computational Methods*, 2(1):21–27, 2012.

[23] Thomas C Sharkey and H Edwin Romeijn. Greedy approaches for a class of nonlinear generalized assignment problems. *Discrete Applied Mathematics*, 158(5):559–572, 2010.

[24] Y. Shi and R. Eberhart. A modified particle swarm optimizer. In *, The 1998 IEEE International Conference on Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence*, pages 69–73, 1998.

[25] Y. Shi and R. C. Eberhart. Empirical study of particle swarm optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation, 1999. CEC 99*, volume 3, page 1950 Vol. 3, 1999.

[26] Frans Van Den Bergh. *An Analysis of Particle Swarm Optimizers*. PhD thesis, University of Pretoria, Pretoria, South Africa, South Africa, 2002.

[27] Z. Wang, R. Yang, and L. Wang. Multi-agent control system with intelligent optimization for smart and energy-efficient buildings. In *IECON 2010 - 36th Annual Conference on IEEE Industrial Electronics Society*, pages 1144–1149, 2010.

[28] Gerhard Weiss. *Multiagent Systems*. Intelligent Robotics and Autonomous Agents. MIT Press, second edition, 2013.

[29] Michael Wooldridge. *An Introduction to MultiAgent Systems*. Wiley, Chichester, U.K, 2nd edition edition, 2009.

[30] Mutsunori Yagiura, Toshihide Ibaraki, and Fred Glover. A path relinking approach with ejection chains for the generalized assignment problem. *European journal of operational research*, 169(2):548–569, 2006.

[31] Mutsunori Yagiura, Takashi Yamaguchi, and Toshihide Ibaraki. A variable depth search algorithm with branching search for the generalized assignment problem. *Optimization Methods and Software*, 10(2):419–441, 1998.

[32] L. B. Zhang, C. G. Zhou, X. H. Liu, Z. Q. Ma, M. Ma, and Y. C. Liang. Solving multi objective optimization problems using particle swarm optimization. In *The 2003 Congress on Evolutionary Computation, 2003. CEC '03*, volume 4, pages 2400–2405 Vol.4, 2003.

[33] B. Zhao, C. X. Guo, and Y. J. Cao. A multiagent-based particle swarm optimization approach for optimal reactive power dispatch. *IEEE Transactions on Power Systems*, 20(2):1070–1078, 2005.

[34] Lale Özbakir, Adil Baykasoğlu, and Pınar Tapkan. Bees algorithm for generalized assignment problem. *Applied Mathematics and Computation*, 215(11):3782–3795, 2010.