PROPUESTA DE SOLUCIÓN DEL PROBLEMA DE COLOCACIÓN DE MÁQUINAS EMPLEANDO LA TÉCNICA DE OPTIMIZACIÓN POR ENJAMBRES DE PARTÍCULAS (PSO)

■ Maria Eugenia Mazzei
email: mmazzei@una.edu.ve
Universidad Nacional Abierta
Carrera Ingeniería de Sistemas
Caracas-Venezuela

Fecha de Recepción: 12 de Julio 2013
Fecha de Aceptación: 25 de Diciembre de 2013

RESUMEN

Existen problemas en el mundo real en donde se requiere emplear el concepto de distancia rectilínea, uno de ellos es la construcción de circuitos impresos, en donde dado un conjunto de *n* puntos se requiere conectarlos con cables que tengan la menor longitud posible, tomando en consideración la distancia horizontal y la vertical. La distancia rectilínea o norma uno d(**x**, **y**) entre dos puntos **x** y **y**, se define como d(**x**, **y**)= $|x_1 - y_1| + |x_2 - y_2|$. El problema que se aborda en este trabajo es el de colocación de una nueva máquina, dada una distribución preexistente de máquinas, tal que la distancia entre la máquina nueva y las fijas sea mínima, empleando la distancia rectilínea. Este problema ha sido tratado empleando programación lineal, bajo ciertas transformaciones. Para resolverlo aplicaremos la técnica de Optimización por Enjambres de Partículas (PSO). Se comparan los resultados con los obtenidos al utilizar programación lineal.

Palabras Claves: colocación de instalaciones, optimización, programación lineal, computación evolutiva, enjambres de partículas.

PROBLEM SOLUTION PROPOSAL PLACEMENT TECHNIQUE USING THE PARTICLE SWARM OPTIMIZATION FOR MACHINE (PSO)

ABSTRACT

In the real world there are problems that require applying the concept of rectilinear distance. One of such problems is the construction of printed circuits, in which a given set of "n" points is required to be connected with cables measuring the least possible length between any two points in the set, taking into account the horizontal and vertical distances. The rectilinear distance or norm "1" d(x,y) between two points x and y is defined as $d(x,y)=|x1 - y1| + |x2 - y2|$. The problem under discussion in the present work, is the arrangement of a brand new machine given a pre-existent arrangement of machines such that the distance between the new and the old machines is minimized when employing the rectilinear distance. This problem has been treated previously using linear programming methods under certain modifications.

In this work, we will apply Particles Swarm Optimization (PSO) methods to solve it, and results will be compared with those obtained when employing Linear Programming Methods.

Key Words: Facility Location, Optimization, Linear Programming, Evolutionary Computing, Swarm Particles.

I. INTRODUCCIÓN

Sea el problema de colocación de una máquina en un plano en donde hay un conjunto fijo de máquinas, de tal manera que se conocen las coordenadas de estas máquinas, pero se requiere determinar las de la nueva máquina a instalar. Se desea minimizar la distancia entre la nueva máquina y las que están fijas en la planta. Para resolver este problema se emplea el concepto de distancia conocida como distancia rectilínea. Aunque existen procedimientos numéricos clásicos para tratar este tipo de problema, se ha logrado con éxito empleando un modelo de programación lineal, dado que además se pueden incluir restricciones adicionales.

En este trabajo se trata el problema de colocación de una nueva máquina empleando el enfoque evolutivo que provee la optimización por Enjambres de Partículas y se comparan los resultados con los obtenidos al aplicar programación lineal. Este artículo se estructura en las siguientes secciones: en la sección 2 se trata la obtención de la solución aproximada en sistemas lineales. En la sección 3 se aborda el problema de colocación de una maquina. En la sección 4, la solución del problema empleando PSO. En la sección 5 se esboza el algoritmo PSO. En la sección 6, la aplicación del algoritmo PSO para resolver el problema de colocación de la maquina y en la sección 7, las conclusiones.

2. SOLUCIÓN APROXIMADA DE SISTEMAS LINEALES

Sea el problema de hallar la solución **x** si existe, o bien hallar una solución **x** aproximada al sistema de ecuaciones:

$$\mathbf{Ax} = \mathbf{b}$$

En donde **A** es una matriz *m x n*, **x** es un vector de dimensión n y **b** es un vector de dimensión *m*. Si el rango de **A** es menor que *n*, una solución aproximada a este sistema, es un vector **x** de dimensión *n*, el cual hace que $\mathbf{Ax} - \mathbf{b}$ sea tan cercano como sea posible al origen en $\Re^m$. Para medir la "cercanía", se utilizará el concepto de norma de un vector **y** en $\Re^m$. La norma de un vector **y**, es una función de valores reales, $\|\mathbf{y}\|$ que satisface las siguientes condiciones:

1. $\|\mathbf{y}\| \geq 0, \|\mathbf{y}\| = 0 \Leftrightarrow y = 0$
2. $\|\alpha\mathbf{y}\| = |\alpha| . \|\mathbf{y}\|$
3. $\|x + \mathbf{y}\| \leq \|x\| + \|\mathbf{y}\|$ (desigualdad triangular)

Definición (Norma 1): $\|\mathbf{y}\|_1 = \sum_{i=1}^{n} |y_i|$

3. PROBLEMA DE COLOCACIÓN DE UNA MÁQUINA

El problema de colocación de instalaciones en una industria es conocido como *Facility Location Problem* y consiste en hallar la distribución óptima de instalaciones, dadas ciertas restricciones. Kloser y Drexl [6] ofrecen una revisión del estado de arte del problema y proponen una clasificación. En particular destacan los modelos continuos, modelos de colocación de redes y modelos de programación entera-mixta y aplicaciones, detallando su formulación. Los métodos empleados desde un principio para la solución de estos problemas fueron los tradicionales de optimización, continua, entera o de redes. En algunos tipos de ellos se ha empleado el análisis de conglomerados (*Cluster Analysis)*, también se han usado heurísticas como el Templado Simulado (***Simulated Annealing***) y Búsqueda Tabú (***Tabu Search***). Pero en estos últimos años se han resuelto distintos tipos de problemas de esta índole aplicando métodos evolutivos, tales como el algoritmo genético y algoritmos basados en inteligencia social. Wadhwa, y V., Garg [10] en su artículo presentan una revisión del problema y se centran en el problema particular de colocación de instalaciones, considerando capacidades (*capacitated location-allocation problem*), bajo la aplicación del algoritmo genético. Por otra parte, Wang, Sun y Shi [11] proponen un algoritmo híbrido para la resolución de un problema particular de ubicación de estaciones, que combina el algoritmo en genético y el Optimización por Enjambres de partículas en función de puntos de demanda y volumen. Los algoritmos evolutivos y los basados en enjambres han sido empleados últimamente en la resolución de diversos tipos de problemas de optimización lineal/no lineal y en particular de optimización combinatoria con mucho éxito. Algunos investigadores han incursionado en la resolución de problemas de optimización con restricciones empleando algoritmos basados en poblaciones, como Michalewicz [7] quien experimentó con el empleo del algoritmo genético en problemas de programación lineal/no lineal. Por otra parte, **Erğdomuş** [3] ha comprobado la eficiencia del PSO en ejemplos de programación lineal que presentan ciclaje, obteniendo mejores resultados al emplear PSO en comparación con el algoritmo genético.

El problema que abordamos fue propuesto por Bazaraa, Jarvis y Sherali [1] para que fuese resuelto empleando un modelo de programación lineal. Se quiere colocar una nueva máquina dado que hay cuatro máquinas distribuidas en un plano. Estas máquinas están ubicadas en las siguientes coordenadas (x_1, x_2): (3,0), (0,-3), (-2,1) y (1,4), ver Figura No. 1. Sean las coordenadas de la nueva máquina: (x_1, x_2), se requiere formular el problema de hallar una ubicación óptima, tal que la suma de las distancias desde la nueva máquina hasta las cuatro restantes sea mínima, utilizando la distancia rectilínea. Por ejemplo la distancia desde (x_1, x_2) a la primera máquina ubicada en (3,0) es $|x_1 - 3| + |x_2|$.

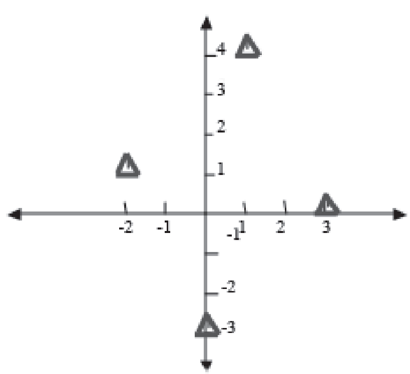

Figura 1: Configuración inicial

La formulación matemática general del problema de aproximación empleando la norma uno es la siguiente:

$$\underset{x \in \Re^n}{\mathrm{Min}} \|Ax - b\|_1 = \sum_{i=1}^{n} |A_i x - b_i| \qquad (1)$$

donde $A_i x = \sum_{j=1}^{m} a_{ij} x_j$

Para formular esta situación como un modelo de programación lineal, es preciso realizar una transformación apropiada que permita manejar el concepto del valor absoluto. La transformación que comúnmente se propone es la siguiente:

$$\underset{x,v}{\mathrm{Min}}\, ev \qquad (2)$$

sujeto a $-v \le Ax \le v$

$$v \ge 0, v \in \Re^m$$

en donde **e** es un vector de *m* componentes iguales a uno:

$$e = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$$

Los vectores de decisión son **v** y **x**, siendo **x** irrestricto en signo. Para operar con estas variables, se propone la siguiente transformación:

$$x = \tilde{x} - \sigma e_n$$

en donde $\mathbf{e}_n$ es un vector de *n* componentes iguales a uno y σ es un escalar,

$$\tilde{x} \ge 0, \ \sigma \ge 0$$

El problema de PL transformado es el siguiente:

$$\underset{x,v}{\mathrm{Min}}\, ev \qquad (3)$$

sujeto a
$$A\tilde{x} - \sigma Ae - v \le b$$
$$A\tilde{x} - \sigma Ae + v \ge b$$
$$v \ge 0, \ \tilde{x} \ge 0, \ \sigma \ge 0$$

Para resolver el problema de programación lineal propuesto, empleando la transformación (3) se definieron la matriz A y el vector b como sigue:

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & -1 \\ -1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, b = \begin{bmatrix} 3 \\ 0 \\ 0 \\ 3 \\ 2 \\ 1 \\ 1 \\ 4 \end{bmatrix}$$

Empleando la formulación descrita, resultaron 11 variables y 16 restricciones. Al resolverlo, se obtuvo como solución óptima: $\tilde{x}_1^* = 1$; $\tilde{x}_2^* = 1$ $v_1^* = 2$; $v_2^* = 1$; $v_3^* = 1$; $v_4^* = 4$; $v_5^* = 3$; $v_6^* = 0$; $v_7^* = 0$; $v_8^* = 3$; $\sigma = 0$ y $z^* = 14$. Por lo tanto las coordenadas en donde estará situada la nueva máquina son: $x_1 = 1$ y $x_2 = 1$. Cabe mencionar, que en este problema existen soluciones alternativas, así podemos hallar otras soluciones óptimas que proporcionan la misma distancia mínima, por ejemplo las coordenadas $x_1 = 0$ y $x_2 = 0$ o también $x_1 = 0$ y $x_2 = -1$.

Existe otra manera de resolver este problema empleando variables complementarias: x_i^+ y x_i^- por cada coordenada, ya que se trata de variables irrestrictas en signo. Por ejemplo para expresar $|x_1 - 3|$ se utiliza la transformación $x_1 - 3 = x_1^+ - x_1^-$. La función objetivo a minimizar es la suma de estas variables complementarias, resultando $z = \sum_{i=1}^{n}\left(x_i^+ - x_i^-\right)$ y las restricciones relacionan las variables complementarias con x_1 y x_2, empleando la transformación descrita. En este problema resultan 18 variables de decisión y 8 restricciones.

4. SOLUCIÓN AL PROBLEMA EMPLEANDO PSO

La optimización por enjambres de partículas (PSO: *Particle Swarm Optimization*) es una técnica heurística de optimización, desarrollada originalmente por James Kennedy y Russell C. Eberhart [4], [5] en el año 1995. Esta técnica se inspira en las bandadas de aves o en los bancos de peces que se desplazan en la búsqueda de alimento. Comparte con los algoritmos evolutivos la generación de poblaciones de forma estocástica y en cierta manera la evolución de nuevas generaciones. Cada individuo de la población se trata como una partícula, se le asocia una posición y velocidad y se mueve en un espacio de búsqueda multidimensional. Las partículas representan las soluciones y se mueven empleando una combinación entre la mejor solución individual (*pbest*) que puede hallar la partícula y la mejor que puede hallar cualquiera de las partículas vecinas (*gbest*). En cada iteración se evalúa el desempeño o *fitness* de cada partícula y se registran las que logran mejor desempeño.

Inicialmente se generan los valores asociados a las partículas aleatoriamente, luego estas partículas se mueven a través del espacio de búsqueda, empleando un sistema de ecuaciones que se modifica en cada iteración, para hallar la mejor solución. Cada partícula se mueve hacia el vecino que tiene más éxito, influyendo sobre las otras partículas. El algoritmo actualiza el enjambre en cada paso, variando la posición y la velocidad de cada partícula, con la aplicación de las siguientes reglas:

$$v_d^{(i)} = wv_d^{(i)} + c_1\varepsilon_1\left(p_d^{(i)} - x_d^{(i)}\right) + c_2\varepsilon_2\left(g_d^{(i)} - x_d^{(i)}\right) \quad (4)$$

$$x_d^{(i)} = x_d^{(i)} + v_d^{(i)} \quad (5)$$

en donde $v_d^{(i)}$ es la d-ésima componente de la velocidad de la partícula *i*, $x_d^{(i)}$ la d-ésima componente del vector posición de la partícula *i*, c_j (j = 1,2) es una constante con valor 2 (obtenido de resultados experimentales) y ε_1 y ε_2 son números aleatorios independientes, uniformemente distribuidos en [0,1], generados en cada actualización, p_d es la d-ésima componente de la mejor actuación de la partícula en su historia, y g_d es la d-ésima componente de la partícula con la mejor posición hallada entre las partículas vecinas, a lo largo de su historia. Esta vecindad se define de acuerdo a la topología del sistema de partículas o tipo de asociación, la cual se establece previamente. El factor ***w*** es conocido como peso inercial y decrece linealmente en cada iteración. El factor c_1 es conocido como el factor de aprendizaje personal o cognitivo y el factor c_2 como el factor de aprendizaje social. Ambos factores tienen mucha influencia en la velocidad de convergencia del proceso de optimización.

El decrecimiento lineal del peso inercial, en cada iteración se lleva a cabo a través de la siguiente expresión:

$$w = W_{max} \; (W_{max} - W_{min}) \; \frac{g}{G} \quad (6)$$

en donde g es el índice de la generación, G es el máximo número de iteraciones preestablecido, w_{max} es un valor mayor que 1, y w_{min} inferior a 0,5. Esta variación del método se ha comprobado que acelera la convergencia, Kennedy [5].

Se han realizado importantes estudios de la convergencia de esta técnica de optimización. Clerc y Kennedy [2] al estudiar las trayectorias de las partículas concluyeron que PSO es sensible a los parámetros tales como los coeficientes de aceleración, peso inercial, tamaño del enjambre y cotas de la velocidad. Bajo estas condiciones el algoritmo converge a la solución óptima o a alguna solución subóptima.

Algoritmo PSO

```
 Comienzo
 Crear población inicial de partículas
{xi} al azar
 Para k = 1 hasta MAXITER hacer
    Para cada partícula i en el enjambre
hacer
     Si (f(x(i)) < f(p(i))  entonces hacer
       Para d = 1  hasta  D  hacer
            p(i)d= x(i)d
       Fin Para
```

```
Fin Si
g = i    // g arbitrario
Para j ∈ J //J: conjunto índices de
                     partículas vecinas
Si f(pj)< f(pg)) entonces g = j
Fin Para
W = Wmax(Wmax – Wmin)*k / MAXITER
Para d = 1 hasta D hacer
v(i)d(t) = w.v(i)d(t)t-1) + c1ε1(pd-x(i)d(t-1))+
        c2ε2(gd- x(i)d(t-1))- x(i)d(t-1))
x(i)d  = x(i)d}  + v(i)d
x(i)d = int(x(i)d
     Fin Para
   Fin Para
Fin Para
Fin
```

5. APLICACIÓN DEL ALGORITMO PSO PARA RESOLVER EL PROBLEMA DE COLOCACIÓN DE LA MAQUINA

Una de las ventajas de esta técnica de optimización es que no se requiere que la función sea diferenciable y ni siquiera continua, tampoco se requiere hacer transformaciones al problema. En el caso del problema propuesto por Bazaraa, Jarvis y Sherali [1], la función de *fitness* natural es la siguiente:

$$z=|x_1-3|+|x_2|+|x_1|+|x_2+3|+|x_1+2|+|x_2-1|+|x_1-1|+|x_2-4|$$

Al aplicar el algoritmo PSO descrito para resolver este problema, se empleó una población de 20 partículas. Cada partícula representa una posición, por lo tanto tiene dimensión dos. La topología de las partículas es la tipo estrella, esto significa que la vecindad de cada partícula es todo el enjambre. Tomando como base las coordenadas de las máquinas fijas, se generaron valores aleatorios iniciales de posición entre [-2,3] para el caso de la componente x_1 y entre [-3,4] para el caso de x_2, luego se transformaron en enteros, ya que por simplicidad se opera en un sistema de coordenadas con componentes enteros. Los valores aleatorios iniciales de la velocidad se tomaron en el intervalo [-4, 5]. Se emplearon factores de aprendizaje c_1 y c_2 con valor 2. La solución óptima obtenida fue: $x_1^*= 0$; $x_2^* = 0$, $z^* = 14$. Luego de aplicar diversas corridas, se obtuvieron otras soluciones óptimas: $x_1^* = 1$; $x_2^* = 0$, $x_1^* = 0$; $x_2^* = 1$, $x_1^* = 1$; $x_2^* = 1$. El número máximo de iteraciones utilizado fue 100. Al resolver este problema empleando Programación Lineal, bajo las transformaciones mencionadas anteriormente, se obtuvieron las mismas soluciones óptimas alternativas, en diversas corridas.

Resultados

Se experimentó con el algoritmo PSO, para un caso de una planta con 12 máquinas. La solución óptima y el valor óptimo de la función objetivo obtenidos se presentan en el cuadro 1. Al resolver el problema empleando programación lineal, una vez formulado de la manera descrita en la sección 3 se obtuvieron 27 variables y 48 restricciones. El número de iteraciones del Método Simplex es 38. La solución óptima y valor óptimo obtenido al aplicar el Simplex, fueron los mismos.

Coordenadas	Solución Optima	Valor óptimo de z
(1,1),(3,2),(7,3),(9,0),(10,3),(12,6)	$x_1^* = 3$; $x_2^* = 3$	68
(3,7),(6,9),(2,8),(1,5),(0,4), (0,2)		

Cuadro 1: Problema con 12 máquinas

Al resolver el problema empleando PSO, la función de *fitness* es la suma de valores absolutos, como en el caso anterior. Se emplearon 20 partículas, las cuales se inicializaron con valores aleatorios en [0,30]. La velocidad de cada partícula fue inicializada con valores aleatorios en [-4,4]. Para el cálculo del peso inercial, se fijaron: $w_{max}= 1{,}5$ y , $w_{min} = 0{,}1$. En el Cuadro 2, se compara el número de variables y restricciones presentes en la formulación de los dos problemas de programación lineal planteados versus el método PSO.

PL	PSO
Primer problema: 11 variables 16 restricciones	Dos matrices: Posición: M ´ 2 (*) Velocidad: M ´ 2
Primer problema: 27 variables 48 restricciones	Dos matrices: Posición: M ´ 2 Velocidad: M ´ 2 Generación de números aleatorios e_1 y e_2 en [0,1]

Cuadro 2: Número de variables requeridas en ambos enfoques (*) M = 20 partículas

En la medida que aumenta el número de máquinas en la planta, aumenta el número de variables y restricciones, en la formulación de PL. En contraste, en el PSO se emplean dos matrices de Mx2 cualquiera sea el número de máquinas.

En cuanto al tiempo de ejecución, se sabe que al aplicar el método Simplex se garantiza la obtención de la solución óptima y se estima un tiempo de ejecución de orden polinomial, en ausencia de ciclaje, mientras que el algoritmo PSO, con una buena selección de parámetros converge a la solución óptima o a alguna solución subóptima. Para la resolución de este tipo de problemas de optimización nos inclinamos hacia el empleo de la técnica PSO, por su simplicidad, por su rápida ejecución y por tratarse de un técnica novedosa que en la actualidad se está experimentando en diferentes tipos de optimización con mucho éxito.

6. CONCLUSIONES

La técnica PSO se basa en un algoritmo sencillo de implementar, que emplea estructuras de almacenamiento simples y es de rápida ejecución. Su uso y aplicaciones se incrementa cada día, siendo actualmente altamente empleada en problemas de optimización de diferente índole. La aplicación del algoritmo PSO en este problema particular, no obliga a hacer transformaciones al problema original, ni requiere que las funciones sean diferenciables. Esto constituye una ventaja sobre la resolución del problema de programación lineal, ya que al incrementar el número de máquinas, aumenta considerablemente el número de variables en la formulación del PPL, mientras que en el caso del PSO se mantiene el mismo número de variables. En la construcción del modelo de programación lineal, por cada máquina deben crearse 4 restricciones. Así, si se tienen 20 máquinas, se deben crear 43 variables y 80 restricciones. En contraste, al aplicar el algoritmo PSO, sólo se construye una función objetivo en forma natural y se puede mantener fijo el número de partículas, cualquiera sea el numero de máquinas. El método halló el valor óptimo en cada corrida. Debido a que la técnica PSO es una heurística, se obtuvieron diferentes soluciones en varias corridas, que en su mayoría eran soluciones alternativas óptimas.

7. BIBLIOGRAFÍA

[1] Bazaraa, M., Jarvis, J., Sherali, H.,D., *Linear Programming and Network Flows*, Wiley \& Sons, Inc, 1990.

[2] Clerc, M., Kennedy, J. *The particle swarm, explosion, stability and convergence in a multidimensional complex space*. IEEE Transactions on Evolutionary Computation, 6(1), 58-73,doi: 101109/4235.985692., 2002.

[3] Ergdomus, P., *Particle swarm optimization performance on special linear programming problems*. Scientific Research and Essays Vol. 5(12), pp. 1506-1518, 18 June, 2010.

[4] Kennedy, J., Eberhart, R. C., *Particle Swarm Optimization*, Proceedings of the 1995 IEEE International Conference on Neural Networks (Perth, Australia): IEEE Service Center, Piscataway, NJ, IV: pp 1942-1948, 1995.

[5] Kennedy, “Particle swarm: social adaptation of knowledge,” in Proceedings of the IEEE Conference on Evolutionary Computation (ICEC ‘97), pp. 303–308, Indianapolis, Ind, USA, 1997.

[6] Kloser, A., Drexl, A., *Facility location models for distribution system design*. European Journal of Operational Research, Vol. 162, pp. 4-29, ISSN 0377-2217, 2005.

[7] Michalewicz, Z., Fogel, D. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 1998.

[8] Parsopoulos, K.E., Vrahatis, M. N., *Particle Swarm Optimization and Intelligence Advances and Applications*, Information Science Reference, IGI Global, 2010.

[9] Shi, Y., Eberhart, R., *A Modified Particle Swarm Optimizer*, Proceedings of the IEEE World Congress of Computing Intelligence, pp. 69 \- 73, 1998.

[10] Wadhwa, V., Garg, D., *Facility Location Problem Using Genetic Algorithm: A Review.* Research Journal of Computer Systems and Engineering - RJCSE ISSN: 2230-8563; e-ISSN: 2230-8571 pp. 63-66 Vol2 Issue 2, 2011.

[11] Wang, L., Sun, X., Shi Z., *New Evolutionary Algorithm Applying to a Type of Facility Location Problem*. Information Technology Journal, 8: 605-609, 2009.