Optimización basada en Mallas Dinámicas. Su aplicación en la solución de problemas de optimización continuos

Amilkar Puris[1], Rafael Bello[2]

Resumen--**En este artículo se presenta una nueva meta-heurística llamada Optimización basada en Mallas Dinámicas (Dynamic Mesh Optimization, DMO), la cual es una técnica de computación evolutiva. Un conjunto de nodos que representan soluciones potenciales de un problema de optimización forma una malla que dinámicamente crece en cada ciclo y se desplaza por el espacio de búsqueda. La formulación de modelo abarca tanto a los problemas de optimización continuos como discretos. En este trabajo se presentan las especificaciones para el caso de problemas continuos y se estudia la aplicación de este nuevo modelo computacional para aproximar algunas funciones conocidas.**

Palabras clave—**computación evolutiva, mallas dinámicas, meta-heurística, optimización continua.**

I. INTRODUCCIÓN

La solución de muchos problemas se puede modelar como un problema de optimización; desde problemas de optimización clásicos como los problemas del Viajante de Comercio (Traveling Salesman Problem, TSP) y de Secuenciación de Tareas (Scheduling, JSP) hasta problemas que se abordan desde la perspectiva de la Inteligencia artificial como el problema de aprendizaje de una Red Neuronal Artificial y el problema de Selección de rasgos caen en esta categoría. La búsqueda de la solución del problema usualmente se plantea como localizar el extremo (mínimo o máximo) de una función objetivo F: $D \rightarrow \Re$; es decir, encontrar un punto $x0 \in D$ tal que:

$F(x0) \leq F(x)$ para todo $x \in D$; para el caso de minimizar la función F, y

$F(x0) \geq F(x)$ para todo $x \in D$; para el caso de maximizar la función F.

El objetivo es asignar valores del dominio, de los permitidos por las restricciones, tal que la función objetivo sea optimizada. Los componentes del problema de optimización son:

- Objetivo – función objetivo (fitness function).
- Conjunto de parámetros (desconocidos) los cuales afectan el valor de la función objetivo.
- Conjunto de restricciones que restringen los valores que se pueden asignar.

Se han desarrollado muchos métodos de optimización global, un aspecto importante de todos estos métodos es su costo computacional, el cual crece no polinomialmente como una función del tamaño del problema [1]. La solución de estos problemas por métodos heurísticos son una alternativa para enfrentarlos [2]. Las técnicas de computación evolutivas pertenecen a esta categoría de métodos. Trabajan sobre un conjunto de soluciones potenciales, llamada población, y realizan la exploración del espacio de búsqueda a través de la cooperación y competencia entre las soluciones potenciales; ejemplos de estas técnicas son los Algoritmos genéticos (AG), y la Optimización basada en partículas (Particle Swarm Optimization, PSO) [3]. También se han denominado modelos computacionales bioinspirados, pues fueron desarrollados a partir de modelos existentes en la naturaleza. Otra técnica de optimización bioinspirada es la Optimización basada en colonias de hormigas (Ant Colony Optimization, ACO). Cuando el espacio de búsqueda es demasiado grande para realizar una búsqueda exhaustiva, las técnicas de búsqueda basadas en población pueden ser una buena alternativa; sin embargo, ellas no garantizan encontrar la solución optima.

Todas estas técnicas son consideradas como meta-heurísticas. Una meta-heurística es una estrategia maestra que guía y modifica otras heurísticas para hallar soluciones a diferentes clases de problemas. Es además, un conjunto de conceptos algorítmicos que pueden ser usados para definir métodos heurísticos aplicables a un conjunto amplio de problemas diferentes. En otras palabras, una meta-heurística puede ser vista como un método heurístico de propósito general designado para guiar una heurística especifica del problema hacia regiones prometedoras del espacio de búsqueda que contengan soluciones de alta calidad; por eso constituye un marco algorítmico general el cual puede ser aplicado a diferentes problemas de optimización con relativamente pocas modificaciones [4].

Los Algoritmos Genéticos (AG) surgen como herramientas para la solución de complejos problemas de búsqueda y optimización, producto del análisis de los sistemas adaptativos en la naturaleza, y como resultado de abstraer la esencia

[1] Dirección del autor 1. E-mail: ayudier@uclv.edu.cu
[2] Dirección del autor 2. E-mail: rbellop@uclv.edu.cu

de su funcionamiento. El término Algoritmo Genético se usa por el hecho de que estos simulan los procesos de la evolución a través del uso de operadores genéticos que operan sobre una población de individuos que "evoluciona" de una generación a otra. Los Algoritmos Genéticos son métodos de búsqueda de propósito general basados en los principios de la genética natural, es decir, son algoritmos de búsqueda basados en los mecanismos de la selección natural y la genética. Un AG puede ser visto como una estructura de control que organiza o dirige un conjunto de transformaciones que se ejecutan iterativamente sobre una población de individuos que representan soluciones para el problema a resolver; estas transformaciones se realizan mediante los llamados operadores genéticos, y considerando una función de calidad o grado de aptitud de un individuo, la cual es una medida que permite comparar las soluciones para determinar cual es mejor [5, 6].

PSO es una técnica de optimización estocástica desarrollada por Eberhart y Kennedy en 1995, la cual se inspira en el comportamiento social de una bandada de pájaros [7, 8]. Es una técnica de optimización basada en una población de partículas generada de forma aleatoria inicialmente; las partículas representan soluciones al problema. Cada partícula mantiene una memoria de su mejor posición, es decir, la que mejor valor de la función objetivo ha alcanzado. Esta posición, así como la mejor solución encontrada por toda la banda, son utilizadas para construir un vector de velocidad para cada partícula en cada ciclo; el vector de velocidad se combina con la posición actual de la partícula para determinar la nueva posición de la misma. De esta forma cada partícula vuela por el espacio de búsqueda. El modelo fue diseñado inicialmente para problemas de optimización continuos, pero posteriormente se han desarrollado diversas adaptaciones del mismo para el caso discreto [9, 10].

Las colonias de hormigas, y más generalmente las sociedades de insectos, son sistemas distribuidos que, a pesar de la simplicidad de sus individuos, presentan una organización social altamente estructurada. Uno de los patrones de comportamiento exhibido por las hormigas es su habilidad para encontrar el camino más corto. Para ello coordinan su actividad mediante una comunicación indirecta lograda por modificación del medio ambiente; esa comunicación se basa solamente en feromonas. El método Optimización basada en colonias de hormigas es un modelo computacional que ofrece las técnicas algorítmicas más exitosas y ampliamente reconocidas basadas en el comportamiento de las hormigas, fue introducido por M. Dorigo [4, 11, 12]. Este modelo asume que la cantidad de feromona sobre un camino es proporcional a la cantidad de hormigas que usaron el mismo en el pasado, y expresa cuan deseado ha sido ese trayecto. Cierta cantidad de feromona se evapora con el tiempo. ACO es una meta-heurística en la cual una colonia de hormigas artificiales coopera en encontrar buenas soluciones a diferentes problemas de optimización discreta. La cooperación es una componente clave de los algoritmos ACO. Una hormiga artificial en ACO es un procedimiento constructivo estocástico que construye incrementalmente una colonia agregando oportunamente componentes a la solución parcial en construcción.

En [13, 14] se ha presentado un nuevo enfoque para implementar la optimización basada en colonias de hormigas para la solución de varias clases de problemas. El mismo consiste en dividir el proceso de búsqueda desarrollado por las hormigas en dos etapas, de modo que en la primera etapa las hormigas encuentran soluciones parciales que son usadas como estados iniciales en la segunda etapa. La base de este enfoque es acercar los estados iniciales a los estados objetivos de la búsqueda. Los resultados experimentales muestran un decrecimiento significativo del tiempo de cómputo.

A. Introducción general del nuevo modelo

Basado en esta idea en este artículo se presenta una nueva meta-heurística llamada Optimización basada en mallas dinámicas (Dynamic Mesh Optimization, DMO), la cual cae en la categoría de las técnicas de computación evolutiva. Un conjunto de nodos que representan soluciones potenciales de un problema de optimización forma una malla que dinámicamente crece y se desplaza por el espacio de búsqueda, para ello en cada ciclo se generan nodos intermedios entre los nodos de la malla y aquellos nodos que son extremos locales (nodos obtenido de mejor valor de la función objetivo, en una vecindad determinada) y el extremo global (nodo obtenido de mejor valor de la función objetivo, en todo el proceso desarrollado); así como a partir de los nodos más externos de la malla. Los mejores nodos de la malla resultante son usados como malla inicial para el ciclo siguiente. La intención del nuevo modelo es permitir una mayor exploración del espacio de búsqueda. La formulación general de la meta-heurística abarca tanto a los problemas de optimización continuos como discretos. En este artículo se propone una formulación específica para el caso de los problemas de optimización continua. Finalmente se estudia el desempeño de la misma para varias funciones conocidas.

Esta meta-heurística comparte elementos comunes con otras, como los Algoritmos genéticos y la Optimización basada en partículas, entre ellos:

- Inicializan una población de soluciones potenciales de forma similar.

- Usan una función de evaluación que determina cuan buena es una solución potencial.
- Son métodos generacionales que repiten el mismo proceso una cantidad determinada de veces.
- Realizan una selección de los mejores individuos para formar nuevas generaciones.

A grosso modo el artículo se presenta con una sección de introducción de algunas meta-heurísticas que inspiraron el desarrollo del nuevo modelo, seguido de una sección 2 que muestra una descripción general de la Optimización Basada en Mallas Dinámicas, posteriormente se hace una especificación del nuevo modelo para el caso continuo. Luego en la sección 4 se realiza un estudio experimental, así como el análisis de los resultados obtenidos, y por ultimo el trabajo presenta una conclusión general.

II. DESCRIPCIÓN GENERAL DE LA META-HEURÍSTICA

La esencia del método DMO propuesto, es crear una malla de puntos en el espacio N dimensional donde se realiza el proceso de optimización de una función FO(x1,x2,…,xn) la cual se mueve mediante un proceso de expansión hacia otras regiones del espacio de búsqueda, pero a la vez se hace más "fina" en aquellas zonas que parecen ser más promisorias. Es dinámica en el sentido que la malla cambia su tamaño (cantidad de nodos) y configuración durante el proceso de búsqueda. Los nodos se representan como vectores de la forma n(x1,x2,…,xn).

En cada ciclo la malla comienza con una cantidad inicial de nodos, y durante el ciclo se realiza un proceso de generación de nuevos nodos hasta alcanzar un número máximo de nodos para la malla. Una vez alcanzada esa cantidad comienza otra iteración usando como malla inicial los mejores nodos de la malla resultante. Durante la búsqueda se conserva el nodo que mejor valor de la función objetivo ha alcanzado hasta ese momento; es decir, ng, denota el extremo global alcanzado hasta ese momento del proceso de búsqueda. El proceso de generación de nodos en cada ciclo comprende los pasos siguientes:

Pasos del proceso de búsqueda desarrollado por DMO:

1. Generación de la malla inicial.
2. Generación de nodos intermedios en dirección a los extremos locales de la malla inicial.
3. Generación de nodos intermedios en dirección al extremo global.
4. Generación de nodos a partir de los nodos más externos de la malla.

El modelo incluye los parámetros:

- Cantidad de nodos de la malla inicial (Ni).
- Cantidad máxima de nodos de la malla en cada ciclo (N), donde 2*Ni ≤ N.
- Cantidad de iteraciones (M).

En el proceso de expansión de la malla en cada ciclo se considera un peso (w), el cual se calcula usando la expresión:

$$w = (w_0 - 0.4) * \frac{M - j}{M + 0.4} \quad (1)$$

Donde w_0 denota el valor de peso inicial (w_0=1.4), y j indica el valor del ciclo actual.

Explicación de cada paso:

Paso 1: Malla inicial de cada ciclo

La malla inicial del primer ciclo cuenta con Ni nodos generados aleatoriamente, mientras que en los siguientes ciclos la malla inicial se forma seleccionando los mejores Ni nodos de la malla resultante de tamaño N en el ciclo precedente.

Paso 2: Generación de nodos intermedios hacia extremos locales

El primer tipo de nodo que se genera en la malla en cada ciclo a partir de los nodos de la malla inicial de ese ciclo tiene por objetivo generar nodos intermedios entre cada nodo y el extremo local más cercano.

Lo primero es calcular para cada nodo sus k-vecinos más cercanos, para ello se utiliza una función de distancia o de semejanza. Luego se selecciona cual de los vecinos es mejor que el nodo, el concepto de mejor depende de si se está maximizando o minimizando la función FO. Si ninguno de los vecinos es mejor que el nodo, entonces este se considera un extremo local y no se generan nodos a partir de él en este paso. En otro caso se genera un nodo (n*) que estará situado entre el nodo (n) y el extremo local (ne). El considerar solamente los vecinos más cercanos en la valoración de lo extremos también fue introducido en el método PSO, como se reporta en [15, 16]; donde al determinar la mejor partícula global se consideran dos enfoques, considerar el mejor de toda la población o la mejor partícula entre los vecinos.

La cercanía del nuevo nodo (n*) al nodo actual (n) o al extremo local (ne) depende de un factor (r), calculado en base a los valores que alcanza la función objetivo FO en el nodo actual y en el nodo extremo local.

Luego se calcula los valores de las componentes del nuevo nodo n* usando la expresión:

$$n^*(i) = f(n(i), ne(i), r) \quad (2)$$

Donde n(i) denota la i-esima componente del nodo n. Mientras mayor sea la diferencia entre los valores de FO en los nodos n y ne, mayor será la cercanía del nodo generado n* al extremo local ne, lo cual será garantizado por el factor r.

Paso 3: Generación de nodos intermedios hacia el extremo global.

Este paso tiene como propósito generar nodos intermedios entre los nodos iniciales de la malla y el extremo global que se tiene hasta ese momento. Para ello se utiliza la expresión:

$$n^*(i) = g(n(i), ng(i), r) \quad (3)$$

Aquí también se utiliza el factor r el cual se calcula a partir de los valores de FO en el nodo y el extremo global ng.

Paso 4: Generación de nodos a partir de los puntos más externos de la malla

En este paso se expande la malla a partir de los nodos más externos de ella (nl), es decir, usando los nodos que se encuentran en la frontera de la malla inicial del ciclo se generan nuevos nodos mediante la expresión:

$$n^*(i) = h(nl(i), w) \quad (4)$$

El valor del peso w se puede calcular usando la expresión (1) y permite que el proceso de expansión sea decreciente durante el proceso de búsqueda, es decir, se produce una mayor expansión al inicio y ésta es casi nula al final.

Nótese que en este paso se seleccionan tantos nodos más externos como sean necesarios para completar el tamaño de la malla en el ciclo; en principio se debe tener N>3*Ni para garantizar que se generen algunos nodos en este paso. Esta etapa garantiza la diversidad en la población.

El proceso de búsqueda realizado por DMO es:

Generación de la malla inicial de forma aleatoria.

Evaluar los nodos de la malla inicial.

Repetir M veces:

Para cada nodo en la malla inicial del ciclo:

Encontrar sus k nodos más cercanos.

Determinar el mejor de los vecinos.

Si el mejor vecino es mejor que el nodo actual, entonces Generar nuevo nodo usando expresión (2).

Para cada nodo en la malla inicial del ciclo generar un nuevo nodo usando la expresión (3).

Repetir hasta completar los N nodos de la malla en el ciclo actual:

Seleccionar el nodo más externo de la malla.

Generar nuevo nodo a partir de este usando la expresión (4).

Construir la malla inicial del siguiente ciclo seleccionando los Ni mejores nodos de la malla final del actual ciclo.

III. ESPECIFICACIÓN DE DMO PARA EL CASO CONTINUO

Para problemas continuos el dominio de todas las variables (x1,x2,…,xn) son los números reales en el intervalo [valor1,valor2], donde valor1 y valor2 son los valores extremos del dominio de cada variable.

Paso 1:

La malla inicial del primer ciclo contará con Ni nodos generados aleatoriamente en el intervalo [valor1, valor2]. Mientras que en los siguientes ciclos la malla inicial se forma seleccionando los mejores Ni nodos de la malla resultante en el ciclo precedente, teniendo en cuenta que la distancia entre ellos (se puede usar la distancia euclidiana) sea mayor que el valor del parámetro cotaDist, el cual establece una distancia mínima entre los nodos; en caso de que la cantidad de nodos seleccionados sea menor que Ni entonces se completa la malla inicial de la etapa mediante la generación aleatoria de nodos.

Paso 2: Función f

Para calcular los k vecinos más cercanos se usa la distancia euclidiana o alguna función de semejanza entre los puntos n y ne.

El factor (r) determinará la cercanía del nuevo nodo al nodo actual (n) o al extremo local (ne), el valor de r se calcula usando una de las expresiones:

Caso Máximo.

$$r = \frac{FO(n)+1}{FO(ne)+1} \quad (5a)$$

Caso Mínimo.

$$r = \frac{FO(ne)+1}{FO(n)+1} \quad (5b)$$

La función f para la generación del nuevo nodo n* generado a partir de cada nodo inicial n de la malla que no sea extremo local y su mejor vecino ne se define por la expresión (6):

$$n^*(i) = \begin{cases} valorM_i, & si \mid valorM_i - ne(i) \mid > cD \ and \ Random \le r \\ ne(i) + w * Random\ [-cD, cD], & si \mid valorM_i - ne(i) \mid \le cD \\ Random\ [valorM_i, ne(i)], & en\ otro\ caso \end{cases} \quad (6)$$

Donde n(i) denota la i-esima componente del nodo n, valorM$_i$ representa el valor medio y se calcula como (n(i)+ne(i))/2, cD es una cota distancia, que esta en dependencia del intervalo de cada variable y de la evaluación de la función objetivo que se este ejecutando, este parámetro evita la convergencia prematura a un valor del dominio de la variable, Random es una función que genera un número aleatorio en el intervalo [0,1] por defecto o entre los valores que se indican como argumento y la variable w se calcula según la expresión 1.

Paso 3: Función g

La generación de nuevos nodos para la malla en el ciclo actual a partir de los nodos iniciales de la malla (n) y el extremo global (ng) se realiza usando las expresiones (7a y 7b, además de la 8):

Caso Máximo

$$r = \frac{FO(n)+1}{FO(ng)+1} \quad (7a)$$

Caso Mínimo

$$r = \frac{FO(ng)+1}{FO(n)+1} \quad (7b)$$

De modo que la función g del modelo en este caso queda definida por la expresión (8):

$$n^*(i) = \begin{cases} vM_i & si \quad Ra[] \le r \\ Ra[vM_i, ng(i)] & en\ otro\ caso \end{cases} \quad (8)$$

Donde n(i) denota la i-ésima componente del nodo n, vM$_i$ valor medio de cada componente i , se calcula como (n(i)+ng(i))/2, y Ra[] es una función que genera un número aleatorio entre los valores vM$_i$ y ng(i).

Paso 4: Función h

En este paso se completa la cantidad total de nodos que debe tener la malla a partir de los nodos más externos de la malla (nl), es decir, los que se encuentran en la frontera de la malla inicial del ciclo. En este caso para determinar los nodos en la frontera de la malla se usa la norma del vector n(x1,x2,…,xn) definida por la expresión (9):

$$\|n\| = \sqrt{n(1)^2 + ... + n(n)^2} \quad (9)$$

Los nodos de mayor y menor norma se consideran como los puntos más externos de la malla. La función h permite generar nuevos nodos mediante las expresiones (10a) y (10b):

Para los nodos en la frontera superior:

$$n^*(i) = nl(i) + w * Rd[] * amplitudMedia_i \quad (10a)$$

Para los nodos en la frontera inferior:

$$n^*(i) = nl(i) - w * Rd[] * amplitudMedia_i \quad (10b)$$

Donde amplitudMedia$_i$ = (|valor1$_i$|+|valor2$_i$|)/2, valor1$_i$ y valor2$_i$ representan los extremos del intervalo del dominio de la variable i.

Si el valor n*(i) no está en el dominio de la variable i; es decir, no pertenece al intervalo [valor1$_i$, valor2$_i$], entonces se corrige usando las siguientes reglas:

Si n*(i)<valor1$_i$ then

n*(i)=Random[valor1$_i$,puntoMedio$_i$]

Si n*(i)>valor2$_i$ then

n*(i)=Random[puntoMedio$_i$,valor2$_i$]

Donde puntoMedio$_i$=(valor1$_i$+valor2$_i$)/2.

IV. VALORACIÓN EXPERIMENTAL DEL MODELO PROPUESTO

El estudio experimental del modelo propuesto se realizó sobre el subconjunto de funciones de prueba f6-f25 de la sesión especial de optimización continua del CEC'05 [17]. Las pruebas se realizaron tanto con 10 como 30 dimensiones para cada función, y se llevaron a cabo 25 repeticiones por cada función. El número máximo de evaluaciones se estableció en 100000 para el caso de 10 dimensiones y 300000 para el caso de 30 dimensiones.

Se comparó el modelo DMO con 3 variantes presentadas (*G-CMA-ES* [18], *DE, K-PC* [19]) como patrón de comparación.

A. Parámetros utilizados

Para los experimentos desarrollados se trabajó con la siguiente configuración del modelo:

El número de nodos que componen la malla inicial Ni=12, tamaño total de la expansión N=42, distancia empleada en el paso 1 para la obtención de los k vecinos más cercano fue la distancia euclidiana, definida por:

$$\sum_{i=1}^{n} \sqrt{(n(i) + na(i))^2}$$, para este proceso se trabajó con k=3, además el valor del parámetro w_0 de la ecuación 1 fue 1.4. Por otra parte la cota de distancia (cD) aplicada en la ecuación 6, se puso en función de las evaluaciones de la función objetivo, y del tamaño del intervalo definido para cada función

objetivo, de esta forma se logró comenzar el proceso de búsqueda con cotas altas y posteriormente disminuir su valor hasta que en la etapa final de la búsqueda se trabajó con cotas pequeñas, este proceso se definió de la siguiente forma:

$$cD = \begin{cases} \frac{|\inf + \sup|}{4}, si\ eA < 15\%cE \\ \frac{|\inf + \sup|}{6}, si\ 15\%cE \le eA < 30\%cE \\ \frac{|\inf + \sup|}{10}, si\ 30\%cE \le eA < 60\%cE \\ \frac{|\inf + \sup|}{20}, si\ 60\%cE \le eA < 80\%cE \\ \frac{|\inf + \sup|}{50}, si\ eA \ge 80\%cE \end{cases}$$

Donde inf y sup representan el valor mínimo y máximo del intervalo de búsqueda para cada función, *eA* evaluación actual del algoritmo y *cE* cantidad de evaluaciones totales definida para cada dimensión.

TABLA I
RESULTADOS EXPERIMENTALES DE LOS ERRORES MEDIOS.

FUNCIONES	DMO	
	10	30
F6	6.34E+000	7.52E+001
F7	1.13E-001 **DK**	5.73E-001
F8	2.03E+001 **D**	2.10E+001 **D**
F9	9.02E+000	8.15E+001
F10	3.31E+000 **D**	6.23E+001 **D**
F11	1.38E+000 **K**	2.50E+001 **DK**
F12	3.15E+001 **DK**	2.33E+004 **G**
F13	5.84E-001 **GDK**	3.17E+000 **DK**
F14	2.13E+000**GDK**	1.16E+001**GDK**
F15	2.46E+002 **DK**	4.23E+002**K**
F16	1.01E+002 **D**	4.84E+001 **DK**
F17	1.76E+002	4.80E+002**GDK**
F18	8.51E+002	8.29E+002**GDK**
F19	4.66E+002 **K**	8.27E+002**GDK**
F20	8.16E+002	8.28E+002**GDK**
F21	4.33E+002**GDK**	5.00E+002**GDK**
F22	7.61E+002	5.35E+002**GDK**
F23	5.59E+002**GDK**	8.32E+002 **K**
F24	4.17E+002	2.12E+002 **DK**
F25	4.12E+002 **D**	2.13E+002 **DK**

B. Presentación de los resultados

En la Tabla I se muestra los errores medios del algoritmo DMO ante cada una de las 25 funciones de prueba, la letra **G** representa que el error medio del algoritmo propuesto es igual o inferior al del algoritmo G-CMA-ES, **D** representa que el error medio del algoritmo propuesto es igual o inferior al del algoritmo DE, y la **K** que el error medio del algoritmo propuesto es igual o inferior al del algoritmo K-PCX. La Tabla II presenta la desviación estándar con respecto al error medio de DMO para cada función de prueba, en este caso no se comparó con los algoritmos mencionados anteriormente porque no fueron presentados por la sesión.

TABLA II
RESULTADOS EXPERIMENTALES PARA LA DESVIACIÓN ESTÁNDAR

FUNCIONES	DMO	
	10	30
F6	5.56E+000	3.20E+001
F7	6.27E-002	1.67E-001
F8	7.34E-002	8.56E-002
F9	4.78E-001	1.11E+001
F10	9.37E-001	1.72E+001
F11	1.01E+000	1.19E+000
F12	1.49E+001	7.71E+003
F13	1.12E-001	4.87E-001
F14	5.84E-001	6.02E-002
F15	1.07E+002	7.62E+001
F16	9.07E-001	1.42E+001
F17	8.62E+000	1.80E+002
F18	3.93E+001	2.41E+000
F19	2.35E+002	1.22E+000
F20	2.28E+001	1.07E+000
F21	9.42E+001	6.17E-004
F22	2.63E+001	7.97E+000
F23	6.40E-006	1.49E+002
F24	5.29E+000	5.35E+000
F25	1.57E+000	3.21E+000

Como se aprecia en la Tabla I, el método DMO obtiene resultados competitivos con respecto a los tres métodos que resultan referencias obligadas en la solución de problemas de optimización continuos. En el caso de las pruebas realizadas para dimensión 10, en más del 63% de las funciones el nuevo método supera a alguno de ellos, mientas que en el 21% se mejora los tres métodos. Para la dimensión 30 se comporta muy superior, ya que en más del 84% de las funciones se logra superar o igualar los resultados de alguno de los algoritmos de referencia, y en el más del 36% se logra superar a los tres. La Tabla II muestra los resultados de la desviación estándar sobre el error medio del algoritmo. Nótese que todas las consideraciones sobre el mejor rendimiento de un algoritmo sobre el otro se hacen únicamente a partir de la información de la que se dispone, que no es más que el error medio de los algoritmos. Sería necesario conocer la distribución de los valores de error de los algoritmos de referencia para poder realizar un estudio mediante tests no paramétricos que nos permitiera dar peso estadístico a dichas afirmaciones.

Es necesario también hacer notar que los resultados alcanzados para la dimensión 30 se obtienen regularmente con la tercera parte de la cantidad definida de evaluaciones de la función objetivo, lo

cual representa que el nuevo método tiene una rápida convergencia a lugares prometedores del espacio de búsqueda.

V. CONCLUSIÓN

En este articulo se ha introducido un nuevo método de búsqueda heurística denominado Optimización basada en mallas dinámicas (Dynamic Mesh Optimization, DMO), el cual se basa en explorar el espacio de búsqueda mediante una malla de nodos que representan soluciones potenciales, esta varia su tamaño dinámicamente mediante la generación de nodos intermedios hacia los extremos locales y el extremo global, así como a partir de los nodos de la frontera de la malla en cada ciclo.

La definición del método permite abordar tanto problemas de optimización continuos como discretos. En este trabajo se ha estudiado el caso continuo. Se realizó un estudio desempeño del método usando un conjunto de funciones estándares clásicas para este tipo de problemas, y se compararon los resultados alcanzados con respecto a tres métodos de referencia en el campo de la optimización continua, los resultados muestran que el método propuesto resulta competitivo para resolver esta clase de problemas.

REFERENCIAS

[1] Parsopoulos, K.E. Stretching techniques for obtaining global minimizers through Particle Swarm Optimization. in Particle Swarm Optimization 2001. Indianapolis, USA.

[2] Michalewicz, Z.y.F., D, How to solve It: Modern heuritics. 2004, Springer.

[3] Bonabeau, E., From natural to Artificial systems, in Swarm Intelligence. 1999: Oxford University Press.

[4] Dorigo, M.a.S.T., Ant Colony Optimization. MIT Press, 2004.

[5] Golberg, D.E., Genetic Algorithms in Search in Optimization and Machine Learning 1998, Addison-Wesley Publishing Company: University of Alabama USA.

[6] Munakata, T. Fundamentals of the new Artificial Intelligence: Beyond traditional paradigms. 1998.

[7] Eberhart, R.C.a.K., J. A new optimizer using particle swarm theory. in Sixth International Symposium on Micromachine and Human Science, Nagoya. 1995. Japan.

[8] Kennedy, J.a.E., R.C. Particle Swarm optimization. in IEEE International Conference on Neural Networks. 1995. NJ.

[9] Bello, P.R.y.G., Two Step Particle Swarm Optimization to Solve the Feature Selection Problem, in International Symposium on Data Analysis, ISDA 2007. 2007: Rio de Janeiro, Brasil.

[10] Sha, D.Y.y.C.-Y.H. A hybrid particle swarm optimization for job shop scheduling problem. in Computers & Industrial Engineering. 2006.

[11] Dorigo, M.e.a. The Ant System: Optimization by a colony of cooperating agents. in IEEE Transactions on Systems, Man and Cybernetics-Part B. 1996.

[12] Dorigo, M.e.a., Ant algorithms for Discrete optimization. Artificial Life, 1999. 5(2): p. 137-172.

[13] Puris Cáceres A.; Bello Pérez, R.T.Y.N.A.M., Y. Two-Stage ACO to solve the Job Shop Scheduling Problem. in 12th Iberoamerican Congress on Pattern Recognition CIARP 2007. 2007. Viña del Mar/Valparaíso, Chile: Springer-Verlag Berlin Heidelberg.

[14] Puris Cáceres.A.; Bello Pérez, R.N.A.M., Y. Two-Step Ant Colony Optimization for solving the Traveling Salesman Problem. in 2nd. INTERNATIONAL WORK-CONFERENCE on the INTERPLAY between NATURAL and ARTIFICIAL COMPUTATION (IWINAC 2007). 2007. España: Springer-Verlag Berlin Heidelberg.

[15] Kennedy, J.a.E., R.C, in Swarm Intelligence. 2001, Morgan Kaufmann

[16] Xiaodong, L., et al. Particle Swarm with speciation and adaptation in a dynamic environment. in GECCO06. 2006. Seattle, Washington, USA.

[17] P. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the cec 2005 special session on real parameter optimization," Tech. Rep., Nanyang Technological University, 2005.

[18] A. Auger and N. Hansen, "A restart cma evolution strategy with increasing population size," in Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2005. 2005, pp. 1769–1776, IEEE Press.

[19] A. Sinha, S. Tiwari, and K. Deb, "A population-based, steadystate procedure for real-parameter optimization," in Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2005. 2005, vol. 1, pp. 514–521, IEEE Press.