OPTIMIZACIÓN DE TRAYECTORIAS DE FRESADO EN CAVIDADES UTILIZANDO EL ALGORITMO LUCIÉRNAGA

JABID QUIROGA [1], EDWIN CÁCERES [2] Y CARLOS PADILLA [3]
1Universidad Industrial de Santander. Escuela de Ingeniería Mecánica. e-mail: jabib@uis.edu.co
2Universidad Industrial de Santander. Escuela de Ingeniería Mecánica. e-mail: edwincaceresramirez@hotmail.com
3Universidad Industrial de Santander. Escuela de Ingeniería Mecánica. e-mail: ing.carlospadilla@outlook.com

Recibido: junio 2014 Recibido en forma final revisado: diciembre 2014

RESUMEN

En este artículo se presenta un método de optimización de rutas de mecanizado en los procesos de desbaste y acabado de cavidades, mediante la utilización del Algoritmo Luciérnaga para encontrar una solución óptima al problema del Agente Viajero (TSP). Para el estudio y optimización de las trayectorias se implementa el algoritmo en el software Matlab®. El proceso de ajuste de los parámetros particulares del algoritmo (población de luciérnagas *n* y número de movimientos *βo*) se realiza a través de análisis estadístico tipo ANOVA. En los resultados se observa que la longitud de la trayectoria de mecanizado obtenida por medio del algoritmo luciérnaga supera el desempeño del software comercial con el cual se compara, lo cual significa una minimización del tiempo de retracción de la herramienta y el tiempo de corte que conlleva la fabricación de cavidades.

Palabras clave: Fresado de Cavidades, Desbaste, Acabado, Optimización de ruta de Mecanizado, Algoritmo Luciérnaga, TSP, CAD/CAM.

OPTIMIZATION OF POCKETS MACHINING USING FIREFLY ALGORITHM

ABSTRACT

In this paper, a route optimization method for roughing and finishing of cavities in machining processes is presented using Firefly Algorithm to find an optimal solution to the Traveling Salesman Problem (TSP). The proposed algorithm is implemented in Matlab® software. The tuning of the firefly parameters (number of fireflies *n* and number of movements *βo*) is performed by a statistical analysis of variance ANOVA. Simulations show a better performance of the firefly algorithm implemented compared with commercial software in minimization the machining trajectory, i.e. retraction and cutting time.

Keywords: Pocket Milling, Rough Cutting, Finish Cutting, Machining Route Optimization, Firefly Algorithm, TSP, CAD/CAM.

INTRODUCCIÓN

En líneas de mecanizado la producción masiva de piezas con cavidades de configuración compleja y múltiples operaciones de maquinado conllevan tiempo que debe ser minimizado buscando mayor producción, rentabilidad y competitividad. Actualmente se disponen de herramientas informáticas, y en particular sistemas CAD/CAM para la generación de trayectorias de mecanizado, que buscan reducir posibles errores y el tiempo de producción de las piezas.

En la literatura se han reportado investigaciones que proponen diferentes enfoques que buscan la reducción de los tiempos de mecanizado en cavidades, disminuyendo el tiempo de corte, el tiempo de retracción de la herramienta y el tiempo de cambio de herramienta.

Tang, Chou & Chen (1998) optimizaron la trayectoria de zigzag usando un algoritmo para la reducción del número de retracciones de la herramienta de corte y dividiendo el área a mecanizar en subregiones. Ahmad, Rahmani & D'Souza (2010) usaron algoritmos genéticos en la selección de una secuencia óptima de herramientas usando fresas de mayor diámetro en el contorno de la pieza y fresas de diámetro menor en el acabado. Zhang & Ge (2009)

propusieron un nuevo enfoque para determinar la secuencia de herramientas de corte óptima para el mecanizado de múltiples funciones en un solo montaje. Ferreira & López (2013) implementaron el uso de operaciones booleanas para la reducción del tiempo de mecanizado mediante la generación de trayectorias de desbaste y acabado usando herramientas de diferentes diámetros.

Por otro lado, se han reportado trabajos en la generación de trayectoria y trayectorias óptimas. Lambregts *et al.* (1996) y Mansor, Hinduja & Owodunni (2006) implementaron algoritmos basados en los diagramas de Voronoi (Subdivisión de un plano en regiones, en donde estas forman lugares más próximos a cada uno de los puntos en un subconjunto específico del plano) para generar trayectorias de mecanizado. Park & Choi (2001) utilizaron el algoritmo de Compensación PWID (Pair-Wise Interference Detection), para generar y optimizar trayectorias de contorno paralelo para el mecanizado de cavidades, eliminado las zonas sin cortar. Held & Spielberger (2009) propusieron un algoritmo que genera trayectorias en espiral para el mecanizado de alta velocidad en cavidades sin islas (zonas no maquinables). Castelino, D'Souza, & Wright (2003) utilizaron un algoritmo heurístico para resolver una variante del problema del agente viajero conocido como GTSP (Generalized Traveling Saleman Problem) para la optimización del tiempo de reacción de la herramienta. Yao & Gupta (2004) propusieron un algoritmo para producir trayectorias de corte optimas mediante la combinación de diferentes patrones de trayectoria como zigzag y contorno paralelo eliminando los tiempos de retracción de la herramienta y el movimiento de reposicionamiento hecho por la herramienta en el proceso de fresado. Suh & Shin (1996) propusieron el uso de una red neuronal tipo SOM (Mapa de auto-organizado por sus siglas en inglés: Self-Organizing Map) para optimizar la generación de trayectorias de mecanizado sin considerar la retracción de la herramienta ni el proceso de acabado.

Por último, existen algunas investigaciones basadas en el consumo de energía relacionado a los distintos esquemas de generación de trayectorias. Pervaiz *et al.* (2012) estudian el consumo de energía para el fresado de cavidades para diferentes tipos de trayectorias generadas a partir de los algoritmos como son: zigzag, espiral de superposición constante, espiral paralela y trayectoria de un solo sentido encontrándose la zigzag como la más eficiente.

En este artículo se utiliza el algoritmo metaheurístico bio-inspirado propuesto por Yang (2008) llamado "*Firefly algorithm*" para plantear el problema del agente viajero (**T**raveling **S**alesman **P**roblem -TSP) como un método para la optimización de trayectorias de fresado en cavidades. El uso de este algoritmo tiene como propósito la reducción de los tiempos de corte y de retracción de la herramienta a través de la generación de trayectorias mínimas y en lo posible continuas de mecanizado para los procesos de desbaste y acabado.

PROBLEMA DEL AGENTE VIAJERO

Según Minetti (2000) el TSP es un problema de optimización combinatoria NP-duro. Este problema se enuncia como la búsqueda del camino más corto de un viajero pasando por *m* ciudades, comenzando en una ciudad determinada y finalizando en la misma ciudad, luego de haber visitado todas ellas solo una vez. Este problema puede ser representado por la Figura 1, cuyos nodos representan cada ciudad, y las líneas la distancia entre un par de ellas, formando de esta manera un ciclo hamiltoniano (es la visita de todos los vértices de un grafo una sola vez, en donde el último vértice visitado es adyacente al primero).

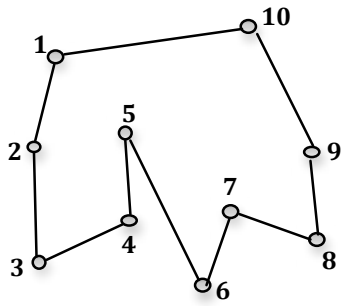


Figura 1. *Presentación gráfica del problema del agente viajero*

El objetivo es encontrar la secuencia de visitas óptimas; con el fin de minimizar la longitud de trayectoria recorrida.

MODELO MATEMÁTICO DEL PROBLEMA AGENTE VIAJERO

En la representación matemática del TSP, la variable c_{ij} es la "distancia" para ir de la ciudad *i* a la ciudad *j* representada en la expresión (1).

$$\textit{El objetivo es minimizar } \sum_{i=1}^{m}\sum_{j=1}^{m} c_{ij}x_{ij} \qquad (1)$$

$$\text{Donde: } x_{ij} = \begin{cases} 1 & \textit{si la ruta incluye la secuencia de ir desde } i \textit{ a } j. \\ 0 & \textit{en otro caso.} \end{cases}$$

Para todo *i* y *j*, la expresión (1) está restringida a:

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad \forall i = 1,\ldots,m \qquad i \neq j \qquad (a)$$

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad \forall j = 1,\ldots,m \qquad i \neq j \qquad (b)$$

$$u_i - u_j + mx_{ij} \leq m - 1 \quad \forall i,j = 1,\ldots,m \;\; i \neq j \qquad \text{(c)}$$

Las restricciones (a), (b) y (c) aseguran que cada x_{ij} sea cero o uno, para que no se repitan visitas a ningún punto. La restricción (a) garantiza un arribo a cada ciudad, mientras que la (b) requiere que una gira incluya una salida desde cada ciudad. La restricción (c) obliga a generar un solo camino y no dos o más caminos disjuntos que cubran conjuntamente todas las ciudades formando "subtours", en donde la variables enteras u_i y u_j indican el orden en que son visitadas las ciudades ej: si u_4=1 y u_6=2, quiere decir que la ciudad 4 es visitada en primer lugar y luego la ciudad 6.

ALGORITMO LUCIÉRNAGA

El Algoritmo Luciérnaga es un algoritmo metaheurístico bio-inspirados propuesto por Yang (2008) en la Universidad de Cambridge. Este algoritmo de búsqueda se basa en el comportamiento poblacional de las luciérnagas y su característica de luminiscencia para comunicarse entre ellas.

El algoritmo luciérnaga está fundamentado en las siguientes reglas, las cuales caracterizan el comportamiento grupal de estas.

Reglas:

- Todas las luciérnagas son asexuales por lo que todas se ven atraídas por todas.

- La atracción es proporcional a su brillo y, dadas dos luciérnagas, la menos brillante se ve atraída por la más brillante; sin embargo, el brillo puede disminuir a medida que la distancia aumenta. Si no existen luciérnagas más brillantes que una luciérnaga dada, esta se mueve aleatoriamente.

- El brillo de una luciérnaga es afectado o determinado por el paisaje de la función objetivo.

En el algoritmo luciérnaga se considera la variación de intensidad de la luz y la formulación de los atractivos. Por simplicidad, siempre se puede asumir que el atractivo de una luciérnaga se determina por su brillo, que a su vez está asociado con la función objetivo codificada. Básicamente, se va comparando la luminosidad de las soluciones generadas tratando de acercar las soluciones que desprenden menos luz hacia las que desprenden un mayor brillo, es decir, acercar las soluciones más lejanas del óptimo hacia las soluciones más cercanas a este. En problemas de minimización, una luciérnaga *x* de mayor luminosidad es la que representa un menor valor dentro de una función objetivo *f(x)*.

El atractivo de una luciérnaga es proporcional a la intensidad de la luz vista por luciérnagas adyacentes, por lo tanto varía según la distancia r_{ij} entre la luciérnaga *i* y la luciérnaga *j* y de las condiciones del ambiente que lo rodea o coeficiente de absorción γ; el atractivo $\beta(r)$ puede ser expresado por cualquier ecuación monótona decreciente, como se observa en la ecuación (1), donde β_o es el atractivo cuando r_{ij}=0.

$$\beta(r) = \beta_0 e^{-\gamma r_{ij}^m}, \quad (m \geq 1) \qquad (1)$$

La distancia entre dos luciérnagas *i* y *j* es la distancia cartesiana en el caso de 2D como se observa en la ecuación (2).

$$r_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \qquad (2)$$

El parámetro γ o coeficiente de absortividad caracteriza la variación de la capacidad de atracción, y su valor es importante en la determinación de la velocidad de la convergencia y de cómo se comporta el algoritmo luciérnaga. El valor de $\gamma \in [0, \infty)$.

El movimiento de una luciérnaga *i* que se siente atraída por otra luciérnaga *j* más atractiva (brillante) está determinado por la ecuación (3):

$$x_i = x_i(1 - \beta) + \beta x_j + \alpha\varepsilon_i \qquad (3)$$

En donde el tercer término representa el movimiento aleatorio de la luciérnaga, con α siendo el parámetro de aleatorización y ε_i corresponde a un vector de números pseudo-aleatorios con distribución uniforme en el intervalo [0, 1]. En nuestro caso ε_i fue reemplazado por *rand-0.5*, donde *rand* es un generador de pseudo-escalares uniformemente distribuidos en [0, 1], atendiendo a lo propuesto por Yang (2008).

En la ecuación (3) pueden suceder los siguientes casos: Si r_{ij} tiende a infinito (la distancia entre la luciérnaga *i* y la luciérnaga *j* es muy grande), entonces β=0 y la luciérnaga *i* se moverá aleatoriamente. Por otro lado, si r_{ij} tiende a cero (la luciérnaga *i* y la luciérnaga *j* están muy cerca), entonces β=1 y la luciérnaga *i* se moverá hacia la luciérnaga *j*. Si β se encuentra entre 0 y 1, entonces la luciérnaga *i* se moverá entre sus alrededores (soluciones) y los alrededores de la luciérnaga *j*. A continuación se muestra el pseudocódigo del algoritmo Luciérnaga propuesto por Yang (2008).

FIREFLY ALGORITHM

Objetive function f(x), x=(x1,.....,xd)^T
Generate initial population of fireflies xi (i=1,2,...,n)
Light intensity Ii at xi is determined by f(xi)
Define light absortion coefficient γ

While *(t<Max Generation)*
  for i=1:n *all n fireflies*
    for j=1:n *all n fireflies (inner loop)*
      if (Ii <Ij),
        Move fireflies i towards j;
      end
    Vary attractiveness with distance r via exp(-γr)
    Evaluate new solutions and update light intensity
  end
end
*Rank the fireflies and find the current global best g**
end
Postprocess results and visualization

DISCRETIZACIÓN DEL ALGORITMO LUCIÉRNAGA PARA EL PROBLEMA DEL AGENTE VIAJERO

En la discretización del algoritmo luciérnaga propuesto por Kusuma & Suyanto (2011), una luciérnaga representa una posible trayectoria o recorrido que da una aproximación a la solución óptima del problema TSP, la cual está conformada por un número de puntos o ciudades las cuales serán visitadas de acuerdo a un orden establecido por los movimientos de las luciérnagas, como se observa en la Figura 2.

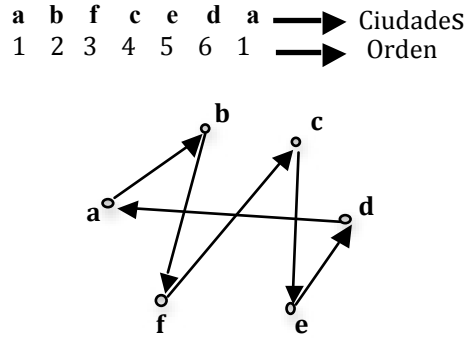


Figura 2. *Representación de la solución de una Luciérnaga*

En el TSP, la función a minimizar es la longitud de la ruta de mecanizado, por consiguiente la luciérnaga más brillante es la de menor longitud de trayectoria, la cual está dada por la distancia euclidiana entre todos los puntos o ciudades visitadas.

Para el algoritmo luciérnaga discreto la distancia entre dos luciérnagas *i* y *j* simbolizada con la letra r_{ij} es la diferencia que hay en el orden de recorrido de las ciudades de la luciérnaga *i* con respecto a la *j*, la cual se define mediante la Ecuación (4) conocida como distancia Hamming.

$$d_{Hammming}(S,T) = \sum_{i=1}^{m} x_k \qquad (4)$$

donde: $x_k = \begin{cases} 0 & si\ S(k) = T(k) \\ 1 & en\ otro\ caso. \end{cases}$

Donde *S* y *T* representan el orden de recorrido de la luciernagas *i* y *j*; *k* representa la posición de ese orden. Esta distancia se determina como el número de veces que las ciudades o puntos coordenados no se encuentran en la misma posición dentro de su orden, como se aprecia en los siguientes ejemplos.

Ejemplo1:	Ejemplo2:
Luciérnaga *i* = [1, 2, 3, 4, 5]	Luciérnaga *i* = [1, 2, 3, 4, 5]
Luciérnaga *j* = [5, 4, 3, 2, 1]	Luciérnaga *j* = [1, 2, 3, 4, 5]
Distancia Hamming=4	Distancia Hamming=0

El resultado de la distancia Hamming se parametriza en un intervalo de *[0-10]* para acotar el valor de r_{ij}; donde $r_{ij}=0$ representará que los puntos o ciudades están siendo recorridos en el mismo orden entre dos luciérnagas y $r_{ij}=10$ representara que los puntos están siendo recorridos en forma totalmente diferente por una luciérnaga con respecto a otra, representada en la Ecuación (5):

$$r_{ij} = \frac{10A}{h} \qquad (5)$$

Donde r_{ij} es la distancia entre dos luciérnagas, *A* es el número total de diferencia entre dos luciérnagas (distancia Hamming) y *h* es el número de ciudades o puntos.

En cuanto al atractivo en la discretización del algoritmo, se adoptó la misma función *β(r)* del algoritmo luciérnaga original (Ecuación (1)) tomando como *m=2* y el parámetro *βo* pasa de ser el atractivo en $r_{ij}=0$ a llamarse Número de Movimientos. Este se determinó a través de un análisis de varianza (tabla 3).

Para determinar el valor del coeficiente de absortividad, (*γ*) se grafica la función atractivo *β(r)* variando la distancia r_{ij} y manteniendo *βo=10*. Este valor de *βo* se escoge debido a que en la experimentación se observó que un valor superior a este, por ejemplo 40, aumenta el tiempo de simulación

drásticamente, y un valor pequeño, por ejemplo 3, limita las opciones en la búsqueda de una solución óptima. La figura 3 presenta los resultados de manera donde se observa un comportamiento monótono decreciente en el intervalo de *0.01* a *0.018*. Basado en los resultados se determinó un valor promedio de γ*=0.015*, valor que permite una buena distribución del número de movimientos βo para las luciérnagas *i* y *j*, evitando que el algoritmo encuentre como solución un óptimo local.

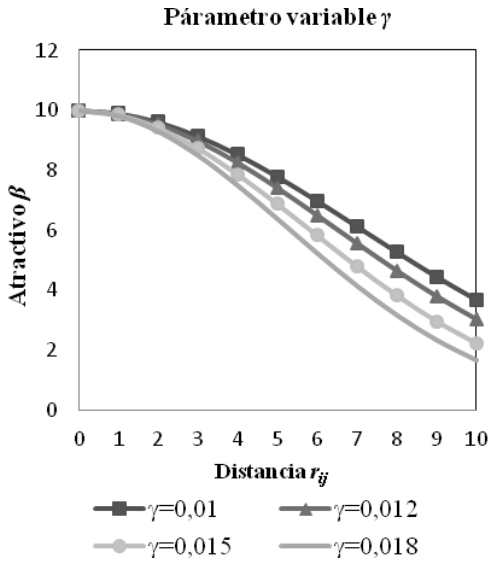


Figura 3. *Comportamiento de la absortividad de luz (γ) frente a la correlación del atractivo y la distancia entre luciérnagas*

En la discretacion de una luciérnaga, los movimientos se realizan por medio de permutación por inversión (Serpell & Smith, 2010). Este tipo de permutación invierte la secuencia de visita de un subconjunto de puntos o ciudades elegida al azar mediante una función llamada *rand()*, que se encarga de escoger 2 posiciones aleatoriamente cualesquiera (*I* y *J*) de un vector de determinado tamaño, las posiciones que se encuentran entre *I* y *J* forman el subconjunto que será reordenado como se observa en el siguiente ejemplo.

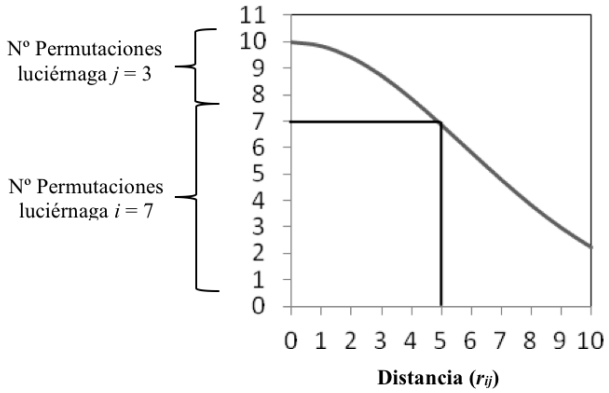
Luciérnaga: $L = [1|2|3|4|5|6|7|8]$

Posiciones escogidas al azar: *I*=2 y *J*=6

Subconjunto de $L = [1|2|3|4|5|6|7|8]$

Luciérnaga: $L' = [1|6|5|4|3|2|7|8]$

La probabilidad que se permute o mueva más veces la luciérnaga *i* o la *j* estará dado por el brillo $\beta(r)$ y la distancia entre luciérnagas como se observa en la Figura 4; de esta forma se acercan las soluciones que desprenden menos luz hacia las que desprenden un mayor brillo.



r_{ij} = Distancia entre Luciérnagas = 5
βo = Número de movimientos = 10

Figura 4. *Distribución de Nº Movimientos de la luciérnaga i y j*

El esquema del pseudocódigo implementado en este trabajo distribuye el número de luciérnagas (preestablecido en el análisis ANOVA), al azar con la función *rand()*, esta función reorganiza las posiciones de los puntos o ciudades de la trayectoria recorrida, representada por cada luciérnaga en la función objetivo. A continuación, basado en el comportamiento de exploración y búsqueda alrededor de la función objetivo (menor distancia de mecanizado), se elegirán la *n* mejores luciérnagas. Se proponen dos métodos que determinan la atracción entre luciérnagas y la exploración del entorno por parte de éstas.

En el primer método, el cual se usa en piezas de configuración simple, las luciérnagas menos brillantes exploran sus alrededores y se mueven hacia las más brillantes con movimientos cortos (permutaciones por inversión con subconjuntos pequeños).

En el segundo método, utilizado en piezas de configuración compleja, por un determinado tiempo las luciérnagas menos brillantes exploran su alrededor con movimientos largos (permutaciones por inversión con subconjuntos grandes) ignorando las luciérnagas que desprenden mayor brillo; después se inicia el movimiento hacia las luciérnagas más brillantes. Al final del proceso de búsqueda cada luciérnaga habrá tenido un número de movimientos (*m*) dados por cada luciérnaga que pertenezca a la población (*n*). Por lo tanto, al final de la iteración habrá (*mxn*) nuevas luciérnagas para empezar la nueva iteración donde sólo se seleccionan las *n* mejores soluciones para el nuevo proceso que se repetirá hasta que se alcance el criterio de parada del algoritmo. A continuacion se muestra el pseudocódigo implementado.

PARÁMETROS DE ENTRADA

-Puntos o ciudades
-Número de Luciérnagas (n)
-Número de Movimientos de las Luciérnagas (βo)
-Número de Iteraciones (G)
-Coeficiente de absorción de Luz (γ)
-Configuración de la figura + Cavidad Simple
+ Cavidad Compleja

PROCESO

```
for k=1:n
 Generar Población de luciérnagas (soluciones Iniciales)
end
Mientras (t<G) o Se cumpla el criterio de Parada
  Función Objetivo: Hallar las n luciérnagas con mayor
  Intensidad de luz (I) , (Trayectorias de menor distancia).
     if Menor Distancia local < Menor Distancia Global
                    SALIDA :Vista Previa de Resultados
                            y Visualización Gráfica
       end
    for i=1:n   Todas las luciérnagas  i
          for j=i:n    Todas las luciérnagas  j
              if Ii < Ij
                 r=Distancia Hamming entre i y j
                 β=βo *exp(-γ*r^2)
                   if t<0.37G && Compleja (Si no se cumple
                             el 37% de las iteraciones)
                       Explorar soluciones alrededor de j
                     else
                      Mover luciérnaga j hacia i
                   end
                   for  u=1:β
                     if  (t<0.27G && Complejo) || Ii = Ij
                        Las Luciérnagas generan movimientos
                       Largos  m veces
                       else
                       Las Luciérnagas generan movimientos
                       Cortos  m veces
                       end
                   end
                   if β~=βo
                        if  t<0.37G && Compleja
                          Explorar soluciones alrededor de j
                             else
                            Mover luciérnaga j hacia i
                        end
                    for  u=β+1:βo
                     if  (t<0.27G && Complejo) || Ii = Ij
                       Las Luciérnagas generan movimientos
                       Largos  m veces
                            else
                        Las Luciérnagas generan movimientos
                       Cortos  m veces
                       end
                    end
                 end
              end
          end
    end
end
```

Resultados y Visualización

Gráficas, distancia y puntos coordenados para los Procesos de Acabado y Desbaste.

En la implementacion del algoritmo algunos parámetros se establecen mediante un análisis de varianza (ANOVA), usando un diseño de experimentos completamente al alzar. Con este análisis estadístico se determinaron los valores óptimos de los factores: "número de luciérnagas" (*n*) y "número de movimientos" (*βo*), para el algoritmo, proponiendo 8 valores dados para cada factor con 5 réplicas o repeticiones de cada uno. El número de réplicas se establece basado en las grandes diferencias entre los factores y su poca variabilidad, Gutiérrez & Vara (2008). Adicionalmente, el bajo número de réplicas resulta del alto costo computacional cuando el algoritmo asume valores grandes de los factores (*n*=12 y *βo* =40); los valores para los factores son los siguientes:

Número de Luciérnagas (*n*): 5, 6, 7, 8, 9, 10, 11 y 12.
Número de movimientos (*βo*): 5, 10, 15, 20, 25, 30, 35 y 40.

Primero se hace un análisis de varianza moviendo el factor *n*, con el factor *βo* estático. Luego se realiza un nuevo análisis de varianza moviendo el factor *βo* con el factor *n* estático, el valor del factor *n* para el segundo análisis de varianza es el valor del tratamiento óptimo que dio en el primer análisis de varianza.

El ajuste de parámetros se hizo para los dos tipos de cavidades: simple y compleja, que tienen las siguientes configuraciones como se muestra en la Figura 5.

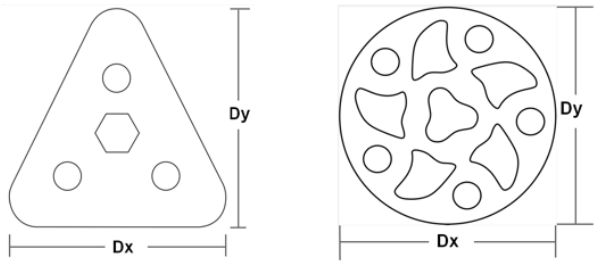


Figura 5. *Cavidad simple y compleja para el ajuste de los parámetros*

- *Cavidad simple*: Contornos de la cavidad sin demasiadas curvas, con pocas islas (zonas interiores al contorno que no deben mecanizarse), sin importar el números de puntos o ciudades para la simulación.

- *Cavidad compleja*: Contornos complicados y puntudos, con muchas islas y puntos o ciudades para la simulación.

En el análisis de varianza se plantean las siguientes hipótesis:

- *Hipótesis nula (Ho)*: los valores de los factores *n* y *βo* no afectan en la respuesta costo y tiempo.

- *Hipótesis Alternativa (H1)*: al menos un par de valores de los factores *n* y *βo* afectan en la respuesta costo y tiempo.

Para parametrizar el factor *n*, se fijó el factor *βo*=10, tanto para la cavidad simple como compleja, por criterio de los autores al compilar el algoritmo, mostrando los resultados en la tabla 1.

Tabla 1. Resultados. Análisis de varianza para el factor *n* para la cavidad simple y compleja.

Cavidad Simple		Cavidad Compleja	
Estadístico de prueba (*Fo*)	Prueba de Fisher (*F*)	Estadístico de prueba (*Fo*)	Prueba de Fisher (*F*)
377,109	2,313	537,283	2,313

Se concluye por la tabla 1 que con una confiabilidad del 95%, el estadístico de prueba es mayor que el valor crítico de la prueba de Fisher (*Fo*>*F*), es decir, que se rechaza la hipótesis nula.

Una vez se cumple la hipótesis alternativa se realiza un análisis de medias por el método de diferencia mínima significativa (**L**east **S**ignificant **D**ifference: LSD), explicado por Gutiérrez & Vara (2008), con el cual se determina el valor óptimo del factor estudiado estadísticamente respecto a la función objetivo. Basado en los resultados del análisis de medias se puede escoger el valor específico del parámetro de acuerdo a su comportamiento frente a la solución. En la Figura 6 y 7 se muestra el comportamiento del factor *n* con respecto a la función objetivo en cavidad simple y compleja.

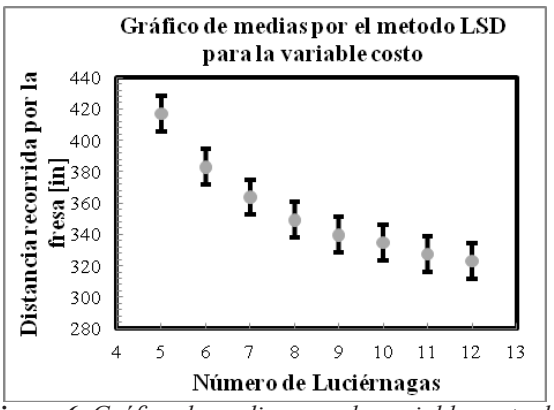


Figura 6. *Gráfico de medias para la variable costo del factor n Cavidad Simple*

De la figura 6, se concluye que los únicos valores del factor *n* que no tienen una diferencia significativa con respecto a la respuesta costo son 11 y 12, para la cavidad simple, es decir que si se usa cualquiera de estos dos valores, la respuesta costo va a hacer prácticamente igual; debido a esto, el valor óptimo para el factor *n* es 11; no se escogió 12 por el tiempo de simulación, ya que el valor 11 del factor *n* tiene un tiempo de simulación mucho más corto que el valor 12.

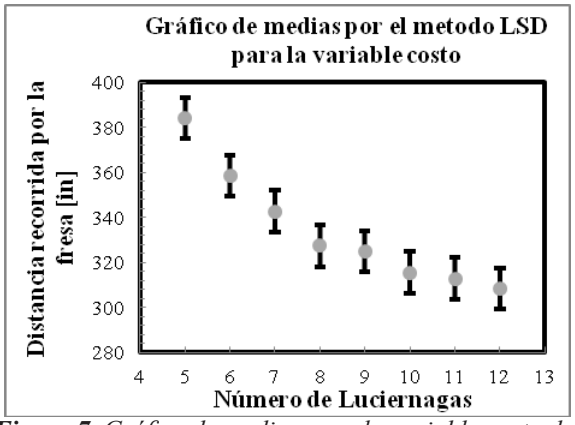


Figura 7. *Gráfico de medias para la variable costo del factor n Cavidad Compleja*

De igual manera se hace la misma conclusión con figura 7; en este caso los valores del factor *n* que no tienen una diferencia significativa son 10, 11 y 12 para la cavidad compleja, siendo el valor óptimo para el factor *n* 10 por el ahorro en el tiempo de simulación.

Hallando los valores óptimos del factor *n*, para las cavidades simples y complejas, se procede hacer el análisis de varianza del factor *βo*, mostrando los resultados de este análisis en la tabla 2; se rechaza nuevamente la hipótesis nula y se vuelve a realizar un análisis de medias para el factor *βo*, como se muestra en las figuras 8 y 9.

Tabla 2. Resultados Análisis de varianza para el factor *βo* para la cavidad simple y compleja.

Cavidad Simple		Cavidad Compleja	
Estadístico de prueba (*Fo*)	Prueba de Fisher (*F*)	Estadístico de prueba (*Fo*)	Prueba de Fisher (*F*)
1582,49	2,313	324,54	2,313

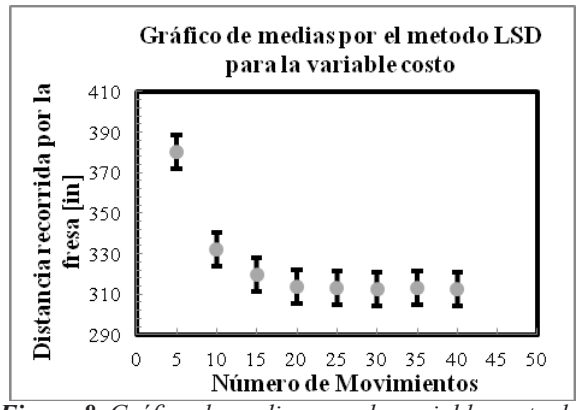


Figura 8. *Gráfico de medias para la variable costo del factor βo Cavidad Simple*

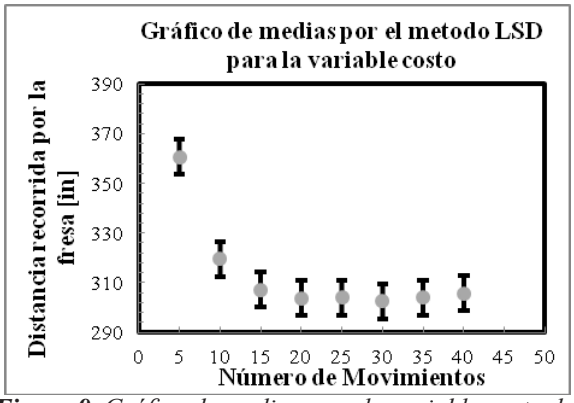


Figura 9. *Gráfico de medias para la variable costo del factor βo Cavidad Compleja*

Se puede concluir de la figuras 8 y 9 que los valores óptimos del factor *βo* son 20 para la cavidad simple y 15 para la cavidad compleja, escogiendo estos valores para el ahorro en el tiempo de simulación.

Los resultados del ajuste de los parámetros mediante el análisis de varianza, se presenta en la Tabla 3.

Tabla 3. Parámetros establecidos por el análisis del ANOVA.

Cavidad Simple	Cavidad Compleja
Nº de Luciérnagas (*n*)=11	Nº de Luciérnagas (*n*)=10
Nº de Movimientos (*βo*)=20	Nº de Movimientos (*βo*)=15

El algoritmo implementado tiene dos criterios de parada, el primero involucra un número preestablecido de iteraciones (entre 2000 y 6000), mientras que el segundo implica la convergencia hacia cero (<1E-15) del cambio de la distancia de mecanización determinada por el algoritmo en 20 iteraciones consecutivas, ver figura 10 y 11.

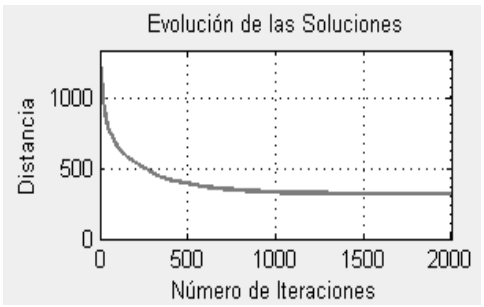


Figura 10. *Curvas de Evolución de las Soluciones para cavidad simple*

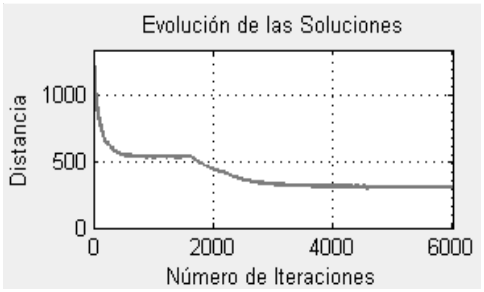


Figura 11. *Curvas de Evolución de las Soluciones para cavidad compleja*

PROCESAMIENTO DE IMAGEN

Para poder detectar los puntos o ciudades que pertenecen a regiones maquinables de una cavidad, en los procesos de desbaste y acabado, se implementó un código en Matlab que permite importar la imagen de la pieza a trabajar en formato JPEG ó BMP. La imagen de la pieza de trabajo se convierte en una matriz de trabajo de Matlab asignando el valor de 1 a los puntos de la pieza maquinables y el valor de 0 los puntos no maquinables. Esta configuración permite determinar dentro de la matriz los puntos o ciudades de trabajo y sus coordenadas(x,y), para ser utilizados en el TSP. En la figura 12, se muestra la generación de puntos realizado por el procesamiento de imagen para el desbaste y el acabado de una cavidad, los círculos representan el diámetro de la fresa a utilizar.

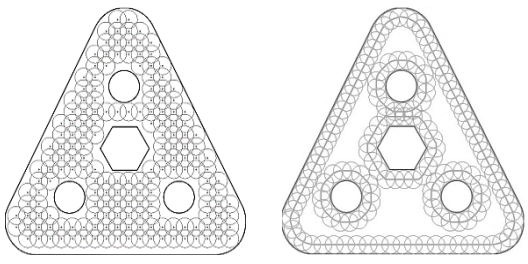


Figura 12. *Graficas de los puntos coordenados de desbaste y acabado*

La distancia entre puntos (*DP*) de la imagen procesada, se determinada en función del diámetro de la fresa (*D*). Para asegurar un mejor desempeño del mecanizado se deben satisfacer los siguientes requerimientos: *DP<(D-0.3D)*, con esta condición se evita la producción de cúmulos o valles, y *DP>(D-0.5D)*, previene un traslape excesivo que produce un retardo en el tiempo de mecanizado. En la figura 13 se puede observar la rejilla cuadrada que se adoptó para la ubicación de los puntos de desbaste y acabado.

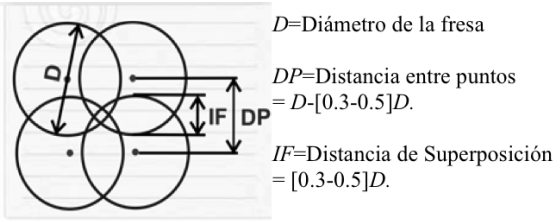


Figura 13. *Rejilla de mecanizado*

En la figura 14 se muestra la representación gráfica de la trayectoria óptima y del área mecanizada para el desbaste, después de aplicar el algoritmo propuesto.

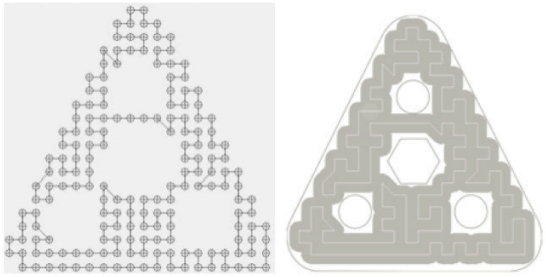


Figura 14. *Representación gráfica de la trayectoria óptima y del área mecanizada para el desbaste*

ANÁLISIS DE RESULTADOS

En esta sección se muestra las pruebas realizadas para piezas con diferentes configuraciones de cavidades entre complejas y simples; el número de puntos o ciudades de trabajo, varían dependiendo de la forma de la cavidad y están entre los 100 y 1200 puntos. En total se simularon y estudiaron 7 piezas, en las cuales se realizaron procesos de acabado y desbaste, ver Figura 15. Para evaluar la efectividad del algoritmo propuesto se compararon las trayectorias óptimas obtenidas con este, bajo las mismas condiciones tanto geométricas de la cavidad y de la herramienta de corte, con el software comercial MasterCam (2010).

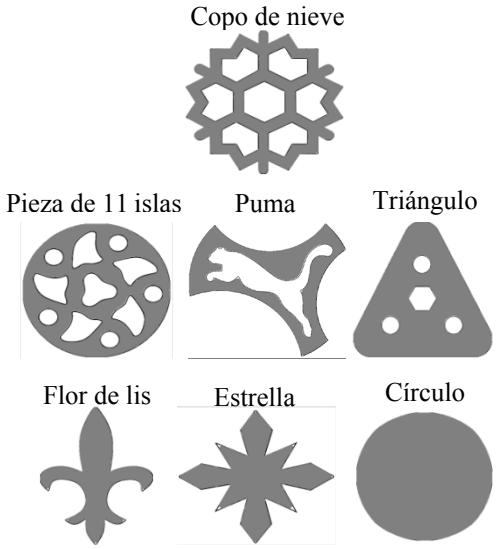


Figura 15. *Resultado final obtenido con el Algoritmo una vez realizado los procesos de acabado y desbaste*

Las pruebas se ejecutaron en un PC con Procesador Intel Pentium Dual-Core T2060 1.60 GHz y una memoria RAM de 1GB tanto para el software comercial MasterCam como para el algoritmo luciérnaga propuesto, codificado en la plataforma de Matlab®.

En las Figura 16, Figura 17 y Figura 18, se encuentran los resultados de la longitud de la trayectoria alcanzada en el proceso de desbaste, acabado y desbaste con acabado respectivamente, obtenidas usando el algoritmo propuesto y el software Comercial MasterCam. Los resultados mostrados usando MasterCam corresponden a la trayectoria obtenida usando el método de Zigzag para todas las cavidades exceptuando a la Flor de Lis y Estrella donde la trayectoria Espiral Paralela presentó mejores resultados.

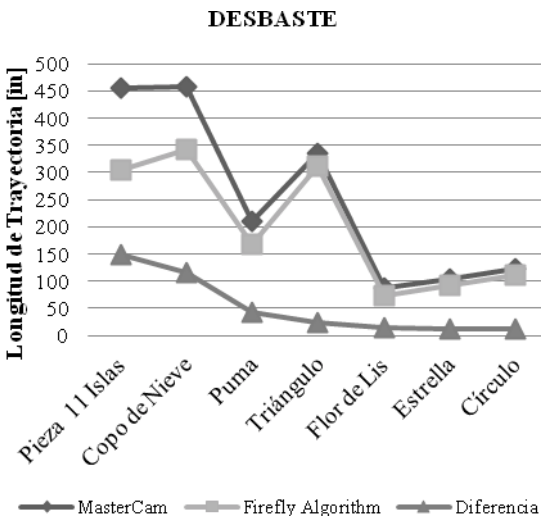


Figura 16. *Longitud de trayectorias de Desbaste*

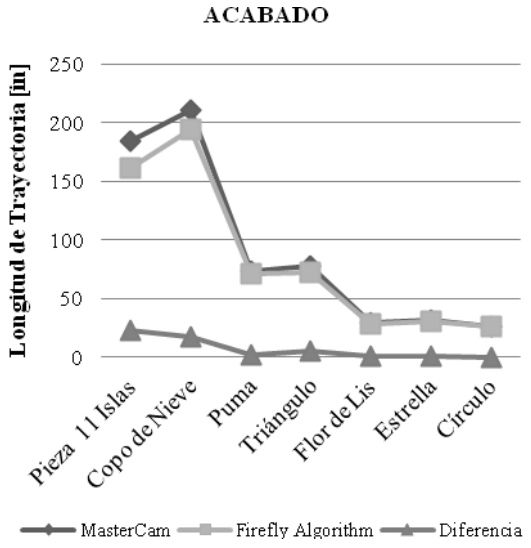


Figura 17. *Longitud de trayectorias de Acabado*

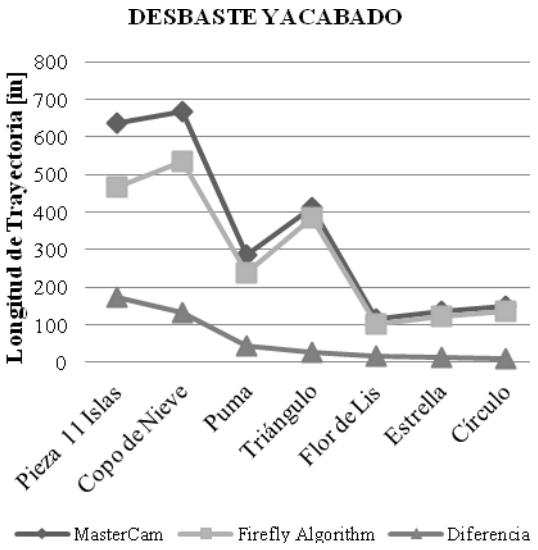


Figura 18. *Longitud de trayectorias de desbaste con acabado*

La longitud de la trayectoria para el proceso de desbaste en el Algoritmo Luciérnaga es menor a la conseguida por MasterCam principalmente en cavidades de configuración compleja y con una gran cantidad de islas.

En el proceso de acabado se observa que el algoritmo propuesto produce una trayectoria menor con respecto a la trayectoria establecida por MasterCam. La diferencia, como en el caso del desbaste, es más significativa en piezas con cavidades con una gran cantidad de islas (zonas no maquinables) y de configuraciones irregulares. En piezas sencillas el algoritmo propuesto y MasterCam se comportan de manera similar.

Al comparar los resultados del algoritmo propuesto con respecto al software comercial se pudo observar que la longitud total de la trayectoria obtenida para los procesos de acabado y desbaste del algoritmo propuesto es menor en todas las cavidades estudiadas, alcanzándose la mayor diferencia, alrededor de 4.36 m, en la pieza de 11 islas.

Para evaluar cuantitativamente el desempeño del algoritmo planteado con respecto al software comercial, se utiliza la Ecuación (6). En las piezas estudiadas se encontró un mejor desempeño por parte del algoritmo luciérnaga, que en el mejor de los casos produjo un 26,94% de mayor desempeño en la pieza con una configuración de 11 islas (figura 19).

$$Desempeño = \frac{Long_{MasterCam} - Long_{alg\ Luciernaga}}{Long_{MasterCam}} \times 100\% \quad (6)$$

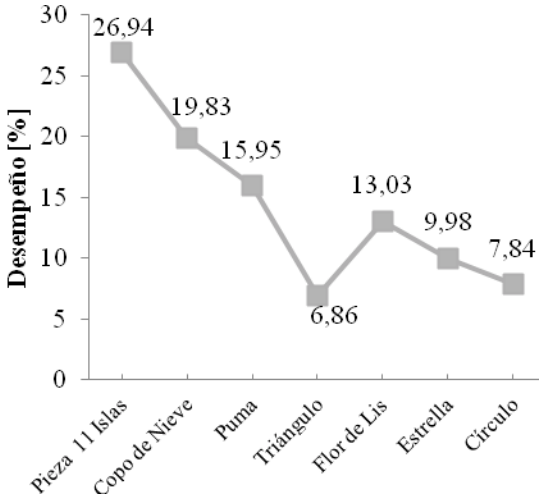


Figura 19. *Desempeño del algoritmo propuesto con respecto al software comercial*

Los resultados anteriores muestran que el algoritmo propuesto, basado en el algoritmo luciérnaga, supera el desempeño del software comercial, sobretodo en cavidades de configuraciones complejas con bastantes islas (zonas no maquinables) en su interior, donde el software comercial produce trayectorias cruzando zonas ya maquinadas o realizando trayectos innecesarios.

Un factor importante a considerar es el tiempo de optimización. Mientras que el software MasterCam encontró las trayectorias de mecanizado en un tiempo no superior a 5 segundos en las piezas estudiadas, el algoritmo propuesto produjo tiempos de ejecución de 3960 segundos en la pieza más compleja. Como se observa en la Figura 20, el tiempo de ejecución para el algoritmo luciérnaga, una vez simulados los procesos de desbaste y acabado, va creciendo en la medida que la pieza presenta una configuración más compleja.

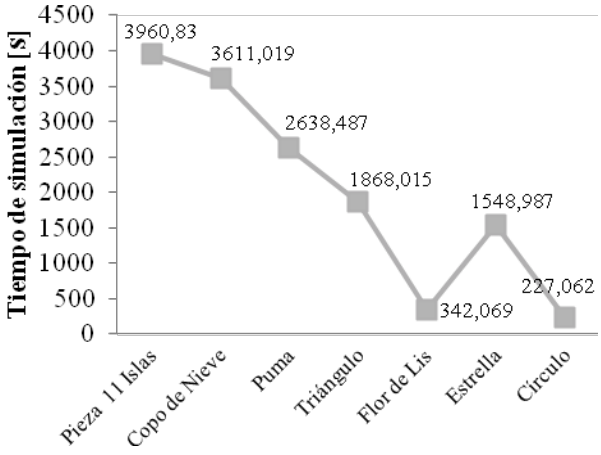


Figura 20. *Tiempo de simulación en segundos del Algoritmo Luciérnaga Discreto*

CONCLUSIONES

En este artículo se propone un algoritmo basado en el metaheurístico algoritmo luciérnaga para la optimización de la trayectoria de mecanizado de piezas con cavidades. Resultados experimentales mostraron que el algoritmo implementado mostró un mejor desempeño en la determinación de la ruta, para los procesos de desbaste y acabado, comparada con la longitud de la ruta dada por un software comercial.

El algoritmo propuesto reduce el tiempo de retracción de la herramienta al producir una trayectoria continua de contorno cerrado, lo cual evita desplazamientos innecesarios de reposicionamiento de la herramienta cuando se realizan varias pasadas para realizar la profundidad de una cavidad.

Aunque el software comercial presenta un menor costo computacional frente al algoritmo luciérnaga implementado, este aspecto no impacta representativamente los costos en la fabricación de piezas con cavidades complejas por cuanto el tiempo de optimización es solo una pequeña fracción del tiempo de fabricación de las piezas, y esta fracción se reduce aún más cuando se van a fabricar un gran número de piezas o cuando estas tienen geometría compleja.

REFERENCIAS

Ahmad Z., Rahmani K., & D'souza R. M. (2010). Applications of genetic algorithms in process planning: tool sequence selection for 2.5-axis pocket machining. J Intell Manuf, 21(4); pp. 461-470.

Castelino K., D'souza R., & Wright P. K. (2003). Toolpath Optimization for Minimizing Airtime During Machining. Journal of Manufacturing Systems, 22(3); pp. 173-180.

Ferreira J. C. E., & Lopez R. A. (2013, Agosto). A Method for Generating Tool Paths for Milling Pockets in Prismatic Parts Using Multiple Tools. 9th International Conference on Automation Science and Engineering, Madison, Wisconsin, United States of America.

Gutiérrez, H., & Vara, R. (2008). Análisis y diseño de Experimentos. (2 ed). México: Editorial McGraw-Hill. 60-90.

Held M., & Spielberger Ch. (2009). A smooth spiral tool path for high speed machining of 2D pockets. Computer-Aided Design, 41(7); pp. 539-550.

Kusuma G., & Suyanto (2011, Septiembre). Evolutionary Discrete Firefly Algorithm for Travelling Salesman Problem. Second International Conference on Adaptive and Intelligent Systems, Klagenfurt, Austria.

Lambregts C.A.H., Delbressine F.L.M., De Vries W.A.H., & Van Der Wolf A.C.H. (1996). An efficient automatic tool path generator for 2.5D free-form Pockets. Computers in Industry, 29; pp. 151-157.

Mansor M. S. A., Hinduja M., & Owodunni O.O. (2006). Voronoi diagram-based tool path compensations for removing uncut material in 2.5D pocket machining. Computer-Aided Design, 38; pp. 194-209.

MASTERCAM X5. (2010). (Versión 14.0.4.33) [Software de computación]. Tolland Connecticut USA: CNC SOFTWARE, INC.

Minetti, G. F. (2000). Una Solución de Computación Evolutiva para el TSP, su posible aplicación en las organizaciones Tesis de Maestría. Universidad Nacional de la Plata, De La Plata, Argentina.

Park S.C., & Choi B. K. (2001). Uncut free pocketing tool-paths generation using pair-wise offset algorithm. Computer-Aided Design, 33(10); pp. 739-746.

Pervaiz S., Deiab, I., Rashid, A., & Nicolescu, M. (2012, Diciembre). An Experimental Analysis of Energy Consumption in Milling Strategies. International Conference on Computer Systems and Industrial Informatics, Sharjah, United Arab Emirates.

Serpell, M., & Smith, J. E. (2010). Self-Adaptation of Mutation Operator and Probability for Permutation Representations in Genetic Algorithms. Evolutionary Computation, 18(3); pp. 491-514.

Suh S. H., & Shin Y. S. (1996). Neural Network Modeling for Tool Path Planning of the Rough Cut in Complex Pocket Milling. Journal of Manufacturing Systems, 15(5); pp. 295-304.

Tang K., Chou S.Y., & Chen L.L. (1998). An algorithm for reducing tool retractions in zigzag pocket machining. Computer-Aided Design, 30(2); pp. 123-129.

Yang, X. S. (2008). Nature-inspired Metaheuristic Algorithm. Universidad de Cambridge, Reino Unido: Luniver Press. 81-96.

Yao Z., & Gupta S.K. (2004). Cutter path generation for 2.5D milling by combining multiple different cutter path patterns. International Journal of Production Research, 42(11); pp. 2141-2161.

Zhang Y., & Ge L. (2009). Selecting optimal set of tool sequences for machining of multiple pockets. Int J Adv Manuf Technol, 42; pp. 233–241.