Optimización mediante Cúmulos de Partículas con Tamaño de Población Variable

Autor: Maria Victoria Leza
Directora: Laura Lanzarini
Jurados: Ramón García Martínez, Claudia Pons, Fernando Tineti

Tesina de Licenciatura en Sistemas
Facultad de Informática
UNIVERSIDAD NACIONAL DE LA PLATA
23 de Julio de 2008

Motivación

- La resolución de **Problemas de Optimización** es de gran interés en la actualidad.

- **Metaheurísticas de Optimización:**

  - Algoritmos Genéticos
    - tamaño de población fija o AG
    - tamaño de población variable o GAVaPS

  - Optimización mediante Cúmulos de Partículas o PSO (Particle Swarm Optimization).

Objetivos de la Tesina

- Presentar una extensión original de PSO, el cual denominaremos VarPSO, que incorpora los conceptos de **edad** y **vecindad** para permitir la **variación del tamaño de la población**.

- Aplicar el método aquí propuesto a la resolución de algunas funciones complejas probando que obtiene mejores resultados que los que habitualmente se logran utilizando población de tamaño fijo.

PSO

- Metaheurística poblacional que simula el comportamiento social de una población de individuos, donde cada individuo intenta mejorar sus conocimientos con interacciones dentro de su entorno social.

PSO

- Consiste en un proceso iterativo y estocástico que opera sobre un cúmulo de partículas.
- Cada partícula del cúmulo representa una posible solución del problema y realiza su adaptación teniendo en cuenta tres factores:
  - su conocimiento sobre el entorno (su valor de fitness)
  - su conocimiento histórico o experiencias anteriores (su memoria o conocimiento cognitivo)
  - el conocimiento histórico o experiencias anteriores de los individuos situados en su vecindario (su conocimiento social)

PSO

- Cada partícula p_i está compuesta por tres vectores y dos valores de fitness:
  - El **vector** $\boldsymbol{x_i} = (x_{i1}, x_{i2}, \dots, x_{in})$ almacena la posición actual de la partícula en el espacio de búsqueda.
  - El **vector** $\boldsymbol{pBest_i} = (p_{i1}, p_{i2}, \dots, p_{in})$ almacena la posición de la mejor solución encontrada por la partícula hasta el momento.
  - El **vector velocidad** $\boldsymbol{v_i} = (v_{i1}, v_{i2}, \dots, v_{in})$ almacena el gradiente (dirección) según el cual se moverá la partícula, donde cada componente del vector debe estar en el rango $[-v_{max}, v_{max}]$.
  - El **valor de fitness** ***fitness_xi*** almacena el valor de aptitud de la solución actual (vector x_i).
  - El **valor de fitness** $\boldsymbol{fitness\_\ pBest_i}$ almacena el valor de aptitud de la mejor solución local encontrada hasta el momento (vector $pBest_i$).

PSO

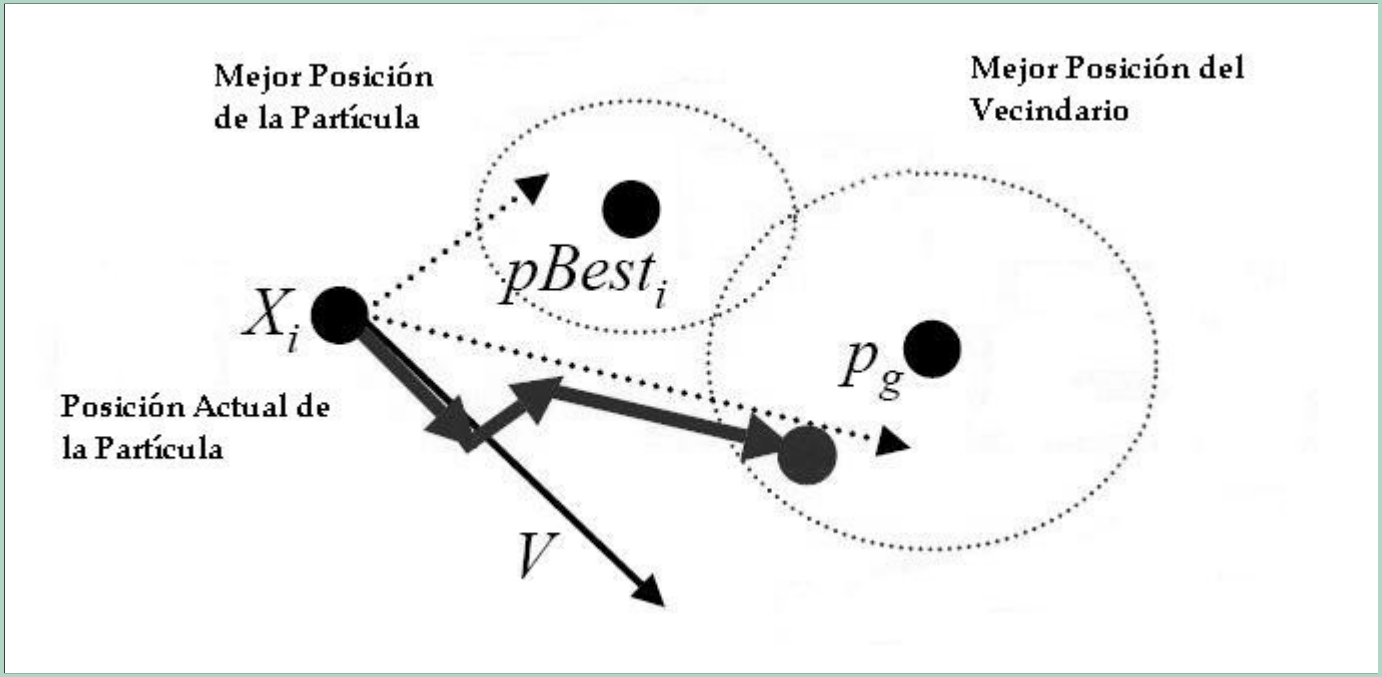
- La posición de una partícula se actualiza de la siguiente forma:

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

- El vector velocidad se modifica teniendo en cuenta su experiencia y la de su entorno:

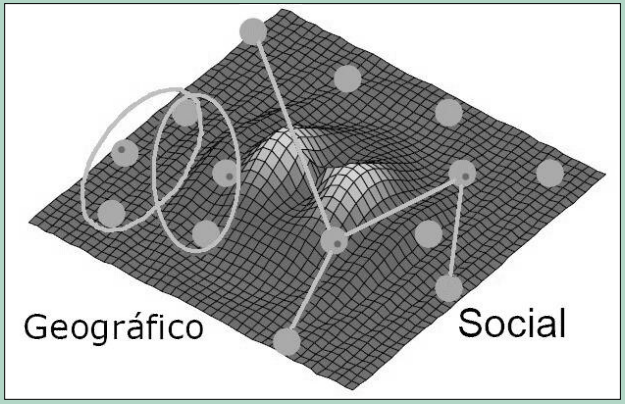
$$v_i(t+1) = \underbrace{w * v_i(t)}_{\text{Recuerdo de la velocidad anterior}} + \underbrace{\varphi_1 * rand_1 * (pBest_i - x_i(t))}_{\text{Influencia cognitiva}} + \underbrace{\varphi_2 * rand_2 * (g_i - x_i(t))}_{\text{Influencia social}}$$

PSO

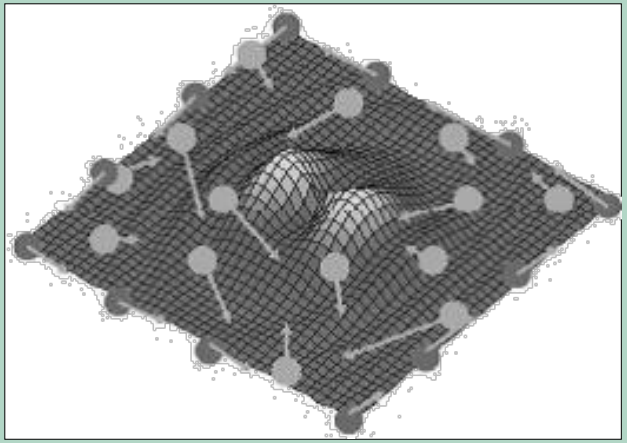

Mejor Posición
de la Partícula
Mejor Posición del
Vecindario
$pBest_i$
X_i
p_g
Posición Actual de
la Partícula
V

Topologías de Cúmulos de Partículas

- **Geográficas:** donde el entorno de cada partícula está formado por aquellos vecinos que se encuentran más cerca de su posición actual.
- **Sociales:** donde el entorno de cada partícula es definido sin importar su posición en el espacio.



Pseudocódigo del Algoritmo PSO



S ← *InicializarCumulo()*
while no se alcance la condición de terminación do
  for *i* = 1 to *size(S)* do
    evaluar la partícula xi del cúmulo S
    if *fitness*(x_i) es mejor que *fitness*($pBest_i$) then
      $pBest_i \leftarrow x_i$; *fitness*($pBest_i$) ← *fitness*(x_i)
    end if
  end for
  for *i* = 1 to *size(S)* do
    Elegir g_i según el criterio de vecindad utilizado
    $v_i \leftarrow w * v_i + \varphi_1 * rand_1 * (pBesti_i - x_i) + \varphi_2 * rand_2 * (g_i - x_i)$
    $x_i \leftarrow x_i + v_i$
  end for
end while
Salida : la mejor solución encontrada

Aspectos Avanzados de PSO

- Valores adaptativos para los coeficientes de aprendizaje cognitivo y social (φ_1 y φ_2), definidos en base a la calidad de la partícula y el entorno.

- Si no se fija una magnitud correcta para v_{max}, la velocidad puede crecer demasiado y ocasionar que las partículas tengan movimientos excesivos. Posible solución:
  - Inercia dinámica (w)

$$w = w_{max} - ((w_{max} - w_{min}) / iter_{max}) * iter$$

- Tamaño del cúmulo.

Algoritmo Genéticos

- Desarrollados por John H. Holland a principios de los años 1960, y se inspiran en la evolución biológica (**Neo-Darwinismo**) y su **base genético-molecular**.

- Forman parte de una familia denominada **Algoritmos Evolutivos**, que incluye las Estrategias de Evolución y la Programación Evolutiva.

- Consiste en hallar los parámetros de los cuales depende el problema, codificarlos en una representación específica, y aplicar los métodos de evolución: selección, reproducción sexual y mutaciones que generan diversidad.

Pseudocódigo del Algoritmo Genético

Generar la población inicial y computar su aptitud;
while (la población no ha convergido) do
    Seleccionar individuos para reproducción;
    Crear descendencia aplicando recombinación y/o mutación a los individuos seleccionados;
    Computar la aptitud de los nuevos individuos;
    Eliminar a los individuos antiguos para hacer lugar para los nuevos cromosomas;
    Insertar la descendencia en la nueva generación;
end while
Salida : la mejor solución encontrada;

GAVaPS

- No usa ninguna de las variantes de selección de AG tradicionales.
- Incorpora el concepto de **Edad** para un cromosoma, depende del fitness del individuo y es equivalente al número de generaciones que el cromosoma ha sobrevivido hasta el momento.
- Otro concepto importante es introducido, el **Tiempo de Vida**, el cual indica el número de generaciones que el individuo puede sobrevivir.

Pseudocódigo de GAVaPS

```
CrearPoblacion(P);
Evaluar el fitness y asignar tiempo de vida a P;
while (la población no ha convergido) do
    Incrementar la edad de cada individuo de P en 1;
    Seleccionar individuos para reproducción;
    Crear descendencia a partir de  los individuos seleccionados;
    Computar la aptitud y tiempo de vida de los nuevos individuos;
    Insertar la descendencia en P;
    Eliminar de P los individuos cuya edad sea mayor a su tiempo de vida;
end while
Salida : la mejor solución encontrada;
```

Estrategias de Asignación de Tiempo de Vida

- Consideraciones necesarias para el cálculo:
  - Soportar individuos cuyo fitness sea mayor al fitness promedio.
  - Ajustar el tamaño de la población actual de acuerdo al paso actual en la búsqueda, tratando de evitar el crecimiento exponencial.
- Estategias:
  - Tradicionales
  - Por clases

Estrategias de Asignación de Tiempo de Vida Tradicionales

- Asignación Proporcional
- Asignación Lineal
- Asignación Bilineal

Estrategias de Asignación de Tiempo de Vida Tradicionales

- Asignación Proporcional

  TiempoDeV ida(Individuo$_i$) = $\min(TV_{Min} + (\eta * Fitness(Individuo_i)) / FIT_{Avg}, TV_{Max})$

  Donde
  - TV_{Min} es el tiempo de vida mínimo que puede tomar $Individuo_i$.
  - FIT_{Avg} es el promedio de todos los fitness de la población.
  - TV_{Max} es el tiempo de vida máximo que puede tomar $Individuo_i$.
  - $\eta = ½ * (TV_{Max} - TV_{Min})$
- Asignación Lineal
- Asignación Bilineal

Estrategias de Asignación de Tiempo de Vida Tradicionales

- Asignación Proporcional
- Asignación Lineal

$$\textbf{TiempoDeVida(Individuo}_i\textbf{)} = TV_{Min} + 2 * \eta * (Fitness(Individuo_i) - ABSFIT_{Min}) / (ABSFIT_{Max} - ABSFIT_{Min})$$

Donde

- $ABSFIT_{Min}$ es el valor absoluto del fitness mínimo de la población.
- $ABSFIT_{Max}$ es el valor absoluto del fitness máximo de la población.

- Asignación Bilineal

Estrategias de Asignación de Tiempo de Vida Tradicionales

- Asignación Proporcional
- Asignación Lineal
- Asignación Bilineal
  - **Si FIT_{Avg} >= Fitness(Individuo$_i$)**

TiempoDeVida(Individuo$_i$) = $TV_{Min} + \eta * (Fitness(Individuo_i) - FIT_{Min}) / (FIT_{Avg} - FIT_{Min})$

  - **Si FIT_{Avg} < Fitness(Individuo$_i$)**

TiempoDeVida(Individuo$_i$) = $1/2 * (TV_{Min} + TV_{Max}) + \eta * (Fitness(Individuo_i) - FIT_{Avg}) / (FIT_{Max} - FIT_{Avg})$

Donde

- FIT_{Min} es el fitness mínimo de la población.
- FIT_{Max} es el fitness máximo de la población.

Estrategias de Asignación de Tiempo de Vida Tradicionales

- Asignación Lineal : mejor performance y un alto costo.
- Asignación Bilineal: menos costosa, pero la performance no es tan buena como en el caso lineal.
- Asignación Proporcional: provee una performance media con un costo medio.

Estrategias de Asignación de Tiempo de Vida por Clases

- Características:
  - Obtiene una distribución adecuada en el rango de valores permitidos, en el sentido de priorizar los individuos que tienen mejor fitness.
  - Limita el crecimiento de la población para encontrar el óptimo.
  - Evita la convergencia al óptimo local.
  - Limita la dispersión de los individuos una vez que el área del óptimo haya sido localizada.

Estrategias de Asignación de Tiempo de Vida por Clases

- Los individuos de la población son agrupados según su valor de aptitud en *k* clases utilizando un método de clustering competitivo del tipo *winner-take-all*.
- Sobre el resultado de este agrupamiento puede aplicarse uno de los siguientes métodos:
  - Asignación de **tiempo de vida fijo** por clase
  - Asignación de **tiempo de vida proporcional** a la cantidad de individuos de cada clase

Asignación de Tiempo de Vida Fijo por Clase

- Se divide el máximo tiempo de vida a asignar por la cantidad de clases (*k*).

- Dentro de una misma clase, sus individuos recibirán un tiempo de vida proporcional a la clase a la que pertenecen y a la cantidad de individuos que se encuentran en su misma clase.

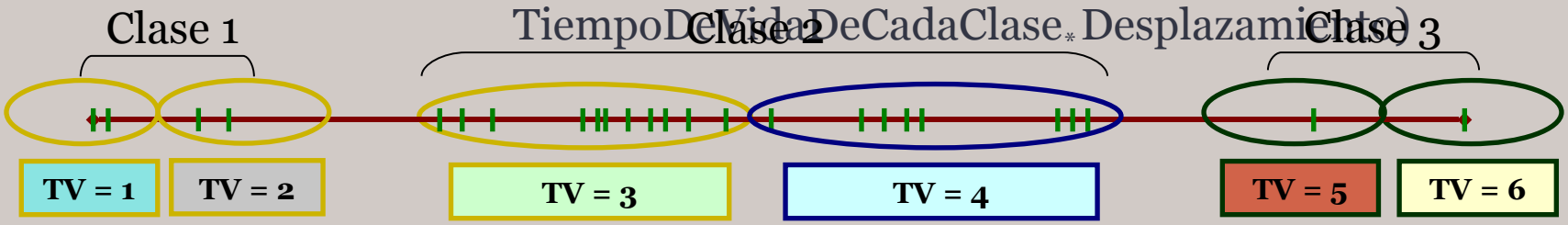
Asignación de Tiempo de Vida Fijo por Clase

TiempoVidaDeCadaClase = MAXIMO_TIEMPO_DE_VIDA/k

TiempoVidaClasesAnteriores = (#ClaseALaQuePertenece - 1) * TiempoVidaDeCadaClase

Desplazamiento = (fitness[i] - Clase[#ClaseALaQuePertenece].MinimoFitness) / abs(Clase[#ClaseALaQuePertenece].MaximoFitness - Clase[#ClaseALaQuePertenece].MinimoFitness)

TiempoDeVida[i] = trunc(TiempoVidaClasesAnteriores + TiempoDeVidaDeCadaClase * Desplazamiento)



Max_Tiempo_Vida = 6

Asignación de Tiempo de Vida Proporcional por Clase

- Cada clase recibe un rango de tiempo de vida proporcional a la cantidad de elementos que contiene.

- Los individuos pertenecientes a las clases numerosas podrán tener un rango de tiempo de vida más amplio.

Asignación de Tiempo de Vida Proporcional por Clase

CantIndivClasesAnt = 0

For i = 1 to #ClaseALaQuePertenece - 1 do

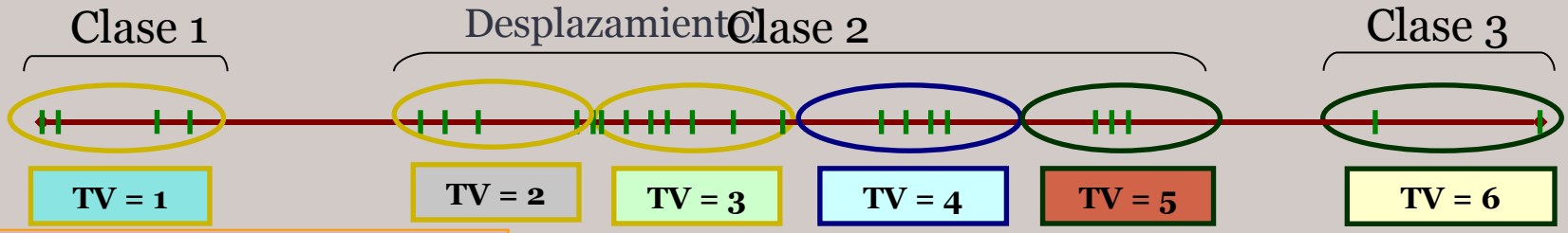
CantIndivClasesAnt = CantIndivClasesAnt + Clase[i].CantidadIndividuos

TiempoVidaClasesAnteriores = MAXIMO_TIEMPO_DE_VIDA * CantIndivClasesAnt / TotalIndividuos

TiempoVidaClaseActual = MAXIMO_TIEMPO_DE_VIDA * Clase[#ClaseALaQuePertenece]. CantidadIndividuos / TotalIndividuos

Desplazamiento = (fitness[i] – Clase[#ClaseALaQuePertenece].MinimoFitness) / abs(Clase[#ClaseALaQuePertenece].MaximoFitness Clase[#ClaseALaQuePertenece].MinimoF itness)

TiempoDeVida[i] := trunc(TiempoVidaClasesAnteriores + TiempoVidaClaseActual *



Max_Tiempo_Vida = 6

Estrategias de Asignación de Tiempo de Vida por Clases

- Aplicar la asignación de tiempo de vida fijo durante un cierto porcentaje de la cantidad de generaciones máximas del algoritmo y en las restantes utilizar asignación de tiempo de vida variable.

- Se requiere de un valor *k* estipulado de antemano, pero este valor depende en gran medida del problema y del rango total de tiempos de vida.

PSO vs AG

PSO	AG
Posee memoria en la optimización	No posee memoria en la optimización
Importa la influencia social del resto de la población	Cada individuo se comporta de manera aislada
No generacional	Generacional

VarPSO

- Objetivo: mejorar la relación de compromiso existente entre la velocidad de convergencia y la diversidad de la población.
  - Permitiendo que las partículas se reproduzcan y mueran en función de su aptitud (**Inserción de nuevos Individuos** y **Tiempo de Vida**).
  - Eliminando los peores individuos de aquellos lugares del espacio de soluciones excesivamente poblados.

Tiempo de Vida

- Determina la permanencia de una partícula dentro de la población.
- Se expresa en cantidad de iteraciones, transcurridas las cuales la partícula es eliminada.
- Tiene una estrecha relación con la aptitud de la partícula y permite que los mejores individuos permanezcan en la población por mayor tiempo, influenciando el comportamiento del resto.
- Para estimarlo, se utilizó el método de asignación de tiempo de vida por clases antes descripto.

Inserción de Partículas

- Objetivos:
  - Incrementar la velocidad de convergencia incorporando individuos en las zonas menos pobladas.
  - Compensar la eliminación de partículas provocada por el cumplimiento de los respectivos tiempos de vida.
- Problema: Determinar los lugares convenientes, dentro del espacio de búsqueda, donde deben insertarse los nuevos individuos, como la velocidad del proceso de inserción.
- Aproximaciones:
  - A través de una **probabilidad de reproducción** .
  - Dividiendo el problema original en dos partes, donde primero se busca **determinar cuántas partículas es necesario incorporar** para luego establecer **dónde deben posicionarse** dentro del espacio de búsqueda.

Inserción: Aproximación 1

- Se analizará el entorno de cada partícula en la población S tomando un radio de distancia r calculado de la siguiente forma:

$$d_i = \min\{|| x_i - x_j || ; \forall j\ x_i, x_j \in S \wedge x_i \neq x_j\} \qquad i = 1..n$$

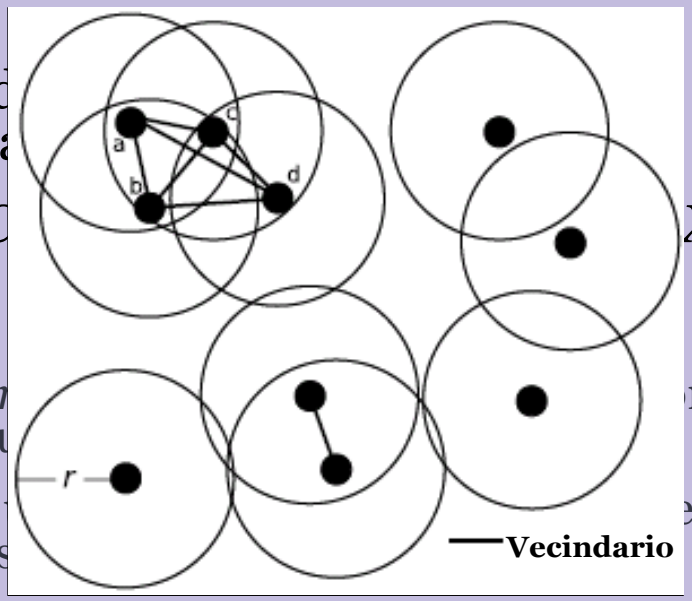
$$r = \Sigma_{i=1..n}\, d_i / n$$

3. La probabilidad d partícula se calcula de la siguiente forma

$$Prob_i = 1 - (C \qquad X\_CANTIDAD)$$

siendo

- *CantVecinosExister* ntradas dentro del entorno del i-ésimo individu
- *MAX_CANTIDAD* epresenta la máxima cantidad de vecinos



Inserción: Aproximación 1

- Cada individuo de la población *S*, **según su probabilidad de reproducción**, puede generar un único descendiente aplicando mutación uniforme sobre la posición actual y copiando en el nuevo individuo su vector velocidad y su conocimiento social.

- **Problema encontrado:** No permite acotar el crecimiento de la población.
  - Al concentrarse los individuos sobre la misma sección del espacio de búsqueda, el radio *r* decrece rápidamente, dando demasiadas posibilidades de reproducción para las partículas del cúmulo, disparando así un crecimiento desmedido.

Inserción: Aproximación 2

- Soluciona el problema planteado por la aproximación 1, ya que divide el problema original en dos partes:
  - Determinar cuántas partículas es necesario incorporar.
  - Establecer dónde deben posicionarse dentro del espacio de búsqueda.

Inserción: Aproximación 2

- Soluciona el problema planteado por la aproximación 1, ya que divide el problema original en dos partes:
  1. Determinar cuántas partículas es necesario incorporar.
     - La cantidad de partículas incorporadas en cada iteración coincide con la cantidad de individuos aislados. Se considera un individuo aislado a aquel que no posea ningún vecino dentro de un radio *r* preestablecido.
  2. Establecer dónde deben posicionarse dentro del espacio de búsqueda.

Inserción: Aproximación 2

- Soluciona el problema planteado por la aproximación 1, ya que divide el problema original en dos partes:
  1. Determinar cuántas partículas es necesario incorporar.
  2. Establecer dónde deben posicionarse dentro del espacio de búsqueda.
     - el 20% de estas nuevas partículas reciben el vector posición de los mejores individuos de la población pero su vector velocidad es aleatorio.
     - el 80% restante es random.

Eliminación de partículas

- Evita la concentración de varias partículas en un mismo lugar del espacio de búsqueda.
- Se evalúan aquellas partículas que poseen vecinos muy próximos, es decir, aquellas cuya distancia al vecino mas cercano es menor que (*r – desviación_media_de_r)*.
- Luego se seleccionan de este conjunto aquellas partículas con fitness más bajo y se eliminan.

Pseudocódigo de VarPSO

```
Pop= CrearPoblacion(N);
CalcularFitness(Pop) y CalcularTiempoDeVidaFijo(Pop);
 w = INER_MAX;
while (no se alcance la condición de terminación) do
    Actualizar la experiencia de cada particula segun  PSO estandar;
    CalcularRadio(Pop, CantHijos, Sentenciados);
    Nuevos = CrearPoblacion(CantHijos);
    Posicionar el 20%de estos nuevos individuos adecuadamente;
    CalcularFitness(Nuevos);
    Eliminar individuos de zonas superpobladas;
    Pop = Pop U Nuevos;
     if (Iter_Actual es mayor que 5 %de las ITER_TOTALES))
         CalcularTiempoDeVidaFijo(Pop); else CalcularTiempoDeVidaVariable(Pop);
     End
     Actualizar los tiempos de vida y eliminar;
     Actualizar dinámica de la inercia;
end while
Salida = la mejor solución encontrada;
```

Problema Abordado

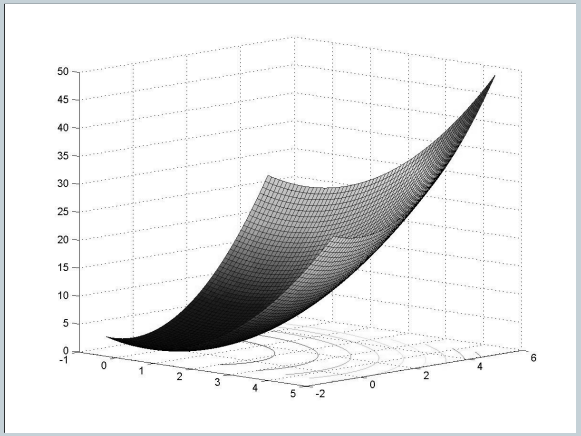
- Objetivo: obtener el valor mínimo de distintas funciones.
- Cada partícula contiene en su vector posición los valores de los argumentos de la función.
- Función de Aptitud de cada partícula o individuo:

$$(c\_max - Valor\_de\_la\_Particula)$$

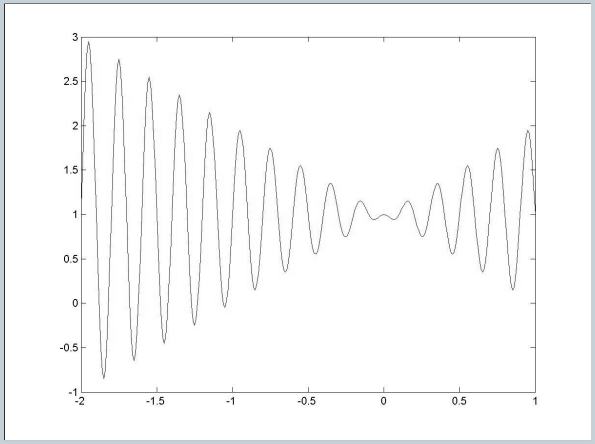
donde c_max representa una cota superior de la función en el intervalo a optimizar y Valor_de_la_Partícula es el resultado de evaluar la función en el vector posición de la partícula correspondiente.

Problema Abordado

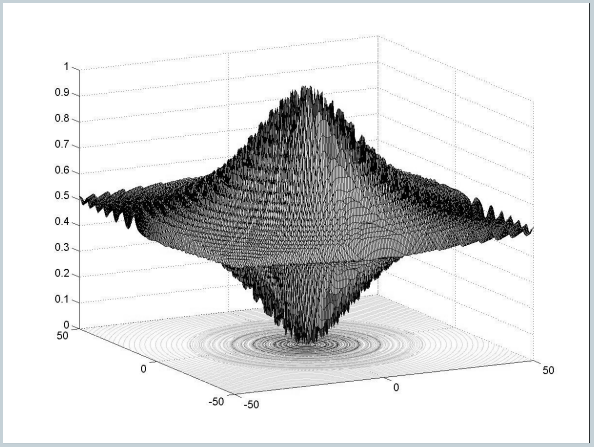
Función F1



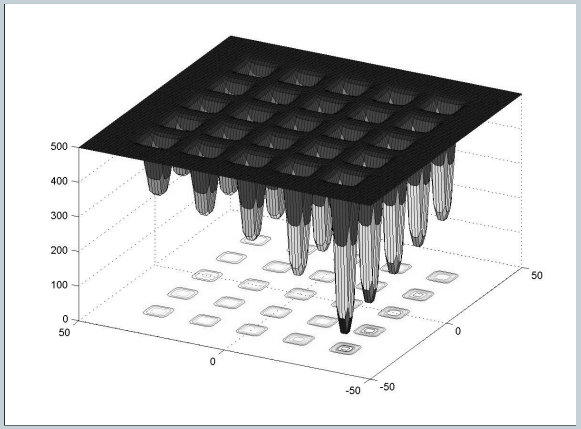
Función F2



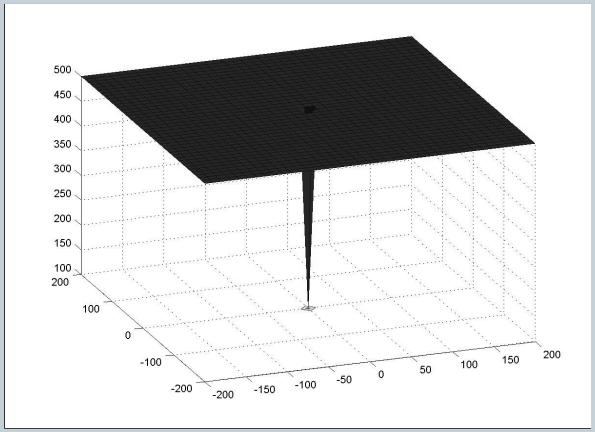
Función F3



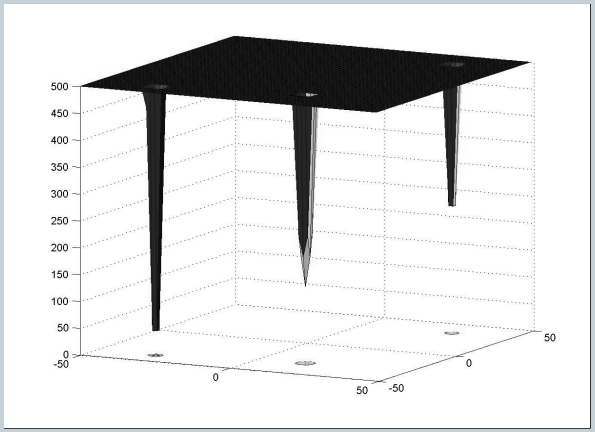
Función F4



Función F5



Función F6



Resultados Obtenidos

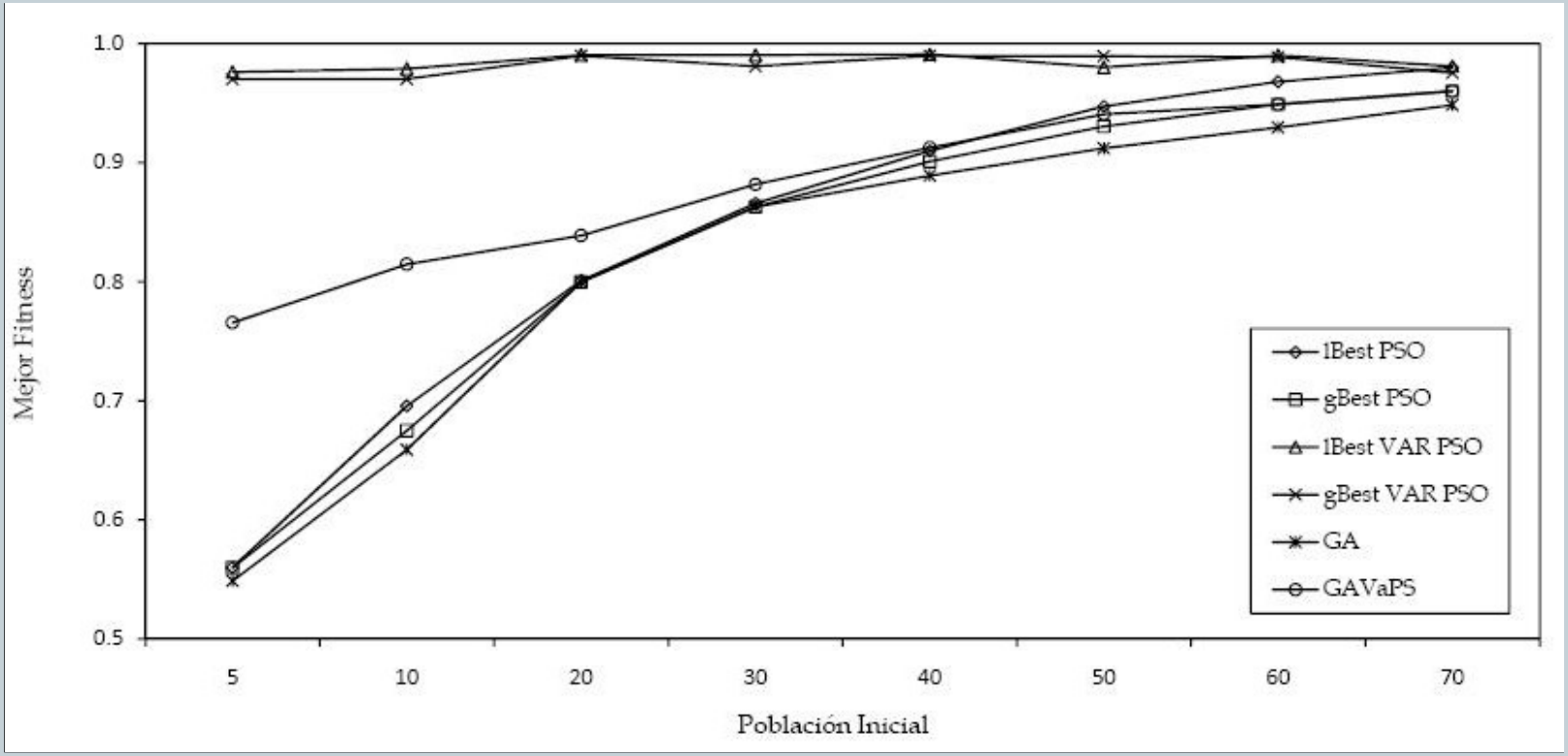
- Para cada función se realizaron 100 pruebas utilizando en cada caso una cantidad máxima de 500 iteraciones.
- Se compararon los resultados obtenidos al aplicar el algoritmo propuesto en esta tesina con el algoritmo PSO de población fija, AG y GAVaPS.
- Se realizaron pruebas con tamaño de población inicial 5, 10, 20, 30, 40, 50, 60 y 70 partículas.

Resultados Obtenidos

- El método propuesto fue superior o igual a la solución basada en población fija y superior a ambas versiones de Algoritmos Genéticos, presentando una mayor diversidad de población.
- El método propuesto utiliza una población inicial inferior a los métodos de población fija.
- En la mayoría de las funciones, el método propuesto realiza menos de la mitad de trabajo que las soluciones con población fija.

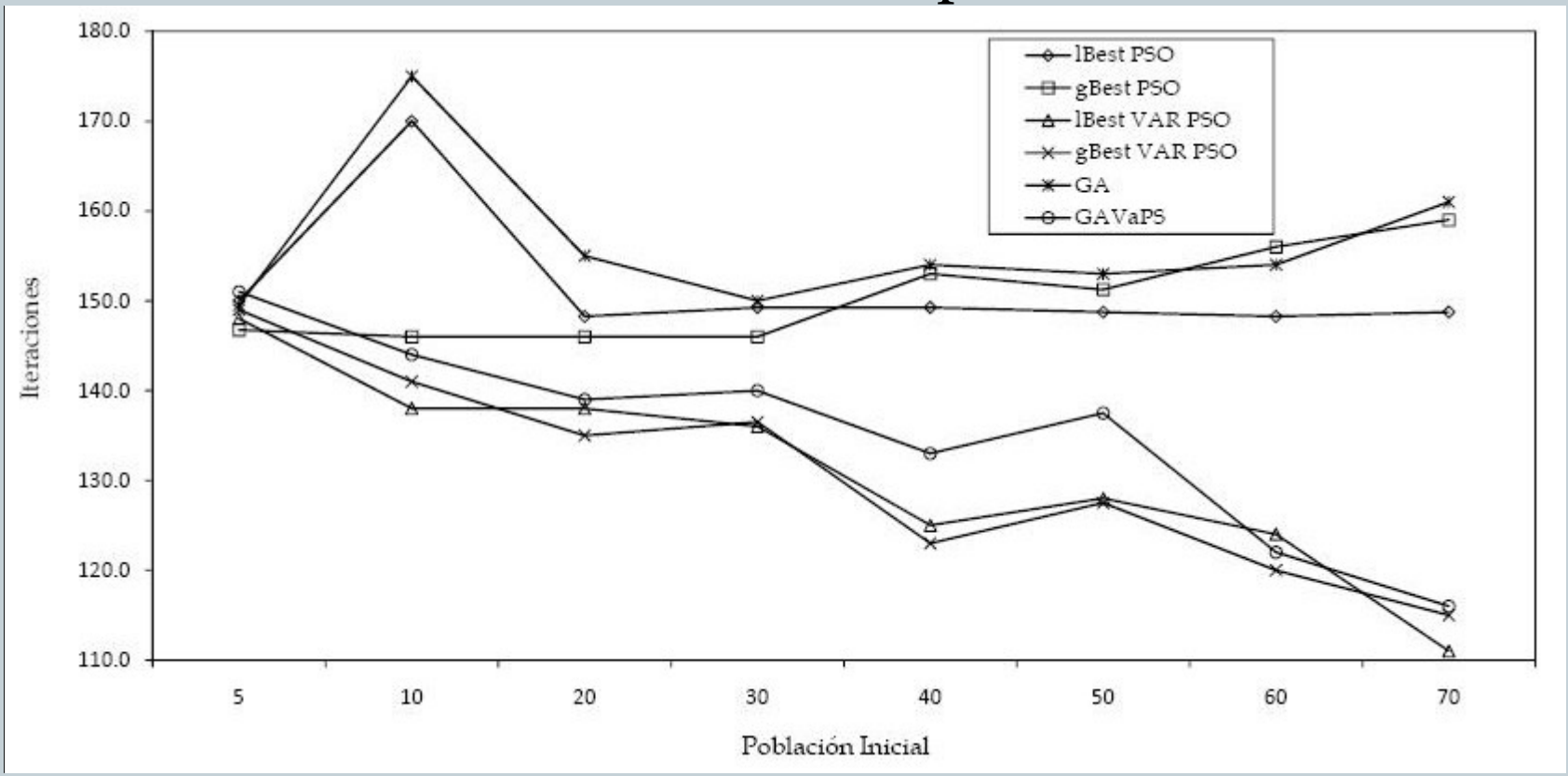
Resultados Obtenidos

Fitness promedio máximo obtenido
para distintos valores de población inicial



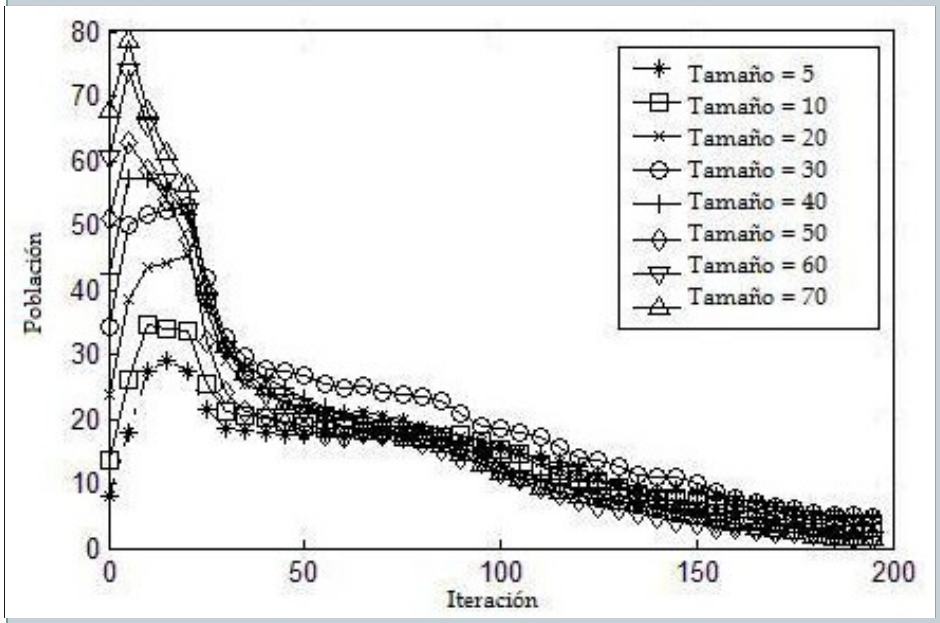
Resultados Obtenidos

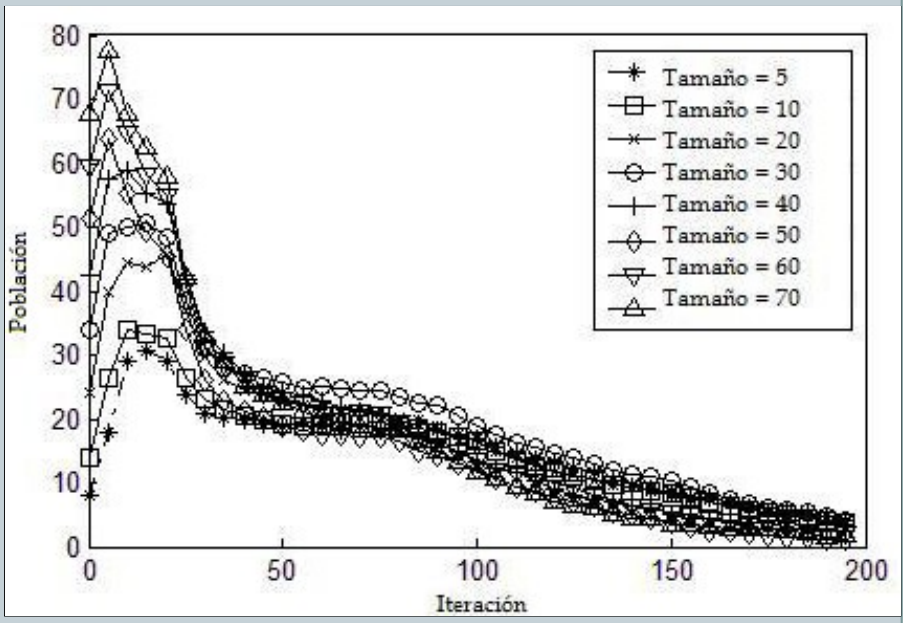
Iteraciones Promedio
en función del tamaño de la población inicial



Resultados Obtenidos

Tamaño promedio de la población utilizando
gBest VAR PSO (izquierda) y lBest VAR PSO (derecha)





Conclusiones

- Se ha profundizado en las metaheurísticas de optimización.
- Se ha ideado una estrategia basada en cúmulos de partículas con tamaño de población variable (VarPSO) basado en los conceptos de edad y vecindario.
- Se diseñó el método de inserción de partículas.
- El otro concepto importante para la variabilidad del tamaño fue el de tiempo de vida basado en asignaciones por clases.
- El algoritmo diseñado, en sus versiones gBest PSO y lBest PSO, se aplicó en diferentes funciones complejas. VarPSO obtuvo un mejor valor de aptitud que las producidas por ambas versiones de PSO con tamaño de población fijo y de Algoritmos Genéticos, tanto de población fija como variable, utilizando una menor cantidad de iteraciones.

Trabajos Futuros

- En virtud de la gran capacidad de exploración demostrada por el método propuesto en esta tesina, se está analizando la posibilidad de aplicar este método a la **adaptación de redes neuronales**.

- Cada partícula representará una red neuronal feedforward completa con arquitectura fija y a lo largo de las iteraciones se buscará adaptar los pesos de conexión.

- Como complejidad de esta aplicación que se presenta en esta tesina, se puede señalar a la medida de similitud que permitirá cuantificar adecuadamente la proximidad entre dos individuos del espacio de búsqueda.

Trabajo Publicado

Bajo el título “Particle Swarm Optimization with Variable Population Size”

en la

Conferencia Internacional en Inteligencia Artificial y Soft Computing 2008 (ICAISC)

FIN

¡MUCHAS GRACIAS POR SU TIEMPO!