UNIVERSIDAD Rafael Belloso Chacín

Revista Electrónica de Estudios Telemáticos

TÉLÉMATIQUE

META-EVOLUCIÓN PARALELA PARA LA ASIGNACIÓN DE PARÁMETROS Y OPERADORES EN ALGORITMOS EVOLUTIVOS

(Parallel meta-evolution for parameters and operators assignment of evolutionary algorithms)

Recibido: 16/10/2013 **Aceptado:** 17/11/2013

Guerra, Aníbal
Universidad de Carabobo, Venezuela
ajguerra@uc.edu.ve

Zaoral, Katherine
Universidad de Carabobo, Venezuela
katherine.zaoral@gmail.com

Rivas, Joel
Universidad de Carabobo, Venezuela
jrivas@uc.edu.ve

RESUMEN

Se presenta una herramienta diseñada para seleccionar operadores (cruce y mutación) y entonar parámetros de control de algoritmos evolutivos (tamaño de la población, probabilidad de cruce y probabilidad de mutación). Esta herramienta fue concebida con un enfoque de meta-evolución: en el nivel superior, un algoritmo genético optimiza a un programa evolutivo, ubicado en el nivel inferior. Dado el volumen de cómputo que esto puede implicar, se implementó una configuración paralela maestro-esclavo, utilizando MPICH2. Los resultados experimentales confirman la validez y generalidad de esta herramienta.

Palabras clave: Meta-evolución, Algoritmos evolutivos, Operadores evolutivos, Parámetros de control.

ABSTRACT

We present a tool for selecting operators (crossover and mutation) and tuning control parameters of evolutionary algorithms (population size, crossover probability and mutation probability). This tool was designed with a focus on meta-evolution: the top level, a genetic algorithm optimizes an evolutionary program in lower level. Given the amount of computation that this may mean is used master-slave parallel configuration, implemented using MPICH2. The experimental results confirm the validity and generality of this tool.

Keywords: Genetic algorithms, Evolutionary programs, Meta-evolution, Control parameters.

Depósito Legal: PPX200002ZU2142 / ISSN:1856-4194. Volumen 13 Edición No 1 – Enero – Junio 2014

INTRODUCCIÓN

Los algoritmos evolutivos (AE) son poderosas herramientas de optimización pertenecientes al grupo de técnicas de la Computación Evolutiva. Para garantizar una convergencia eficiente de estos algoritmos, es conveniente seleccionar los operadores genéticos (cruce y mutación) específicos para cada problema y determinar los valores adecuados para parámetros de control como tasas de cruce (pc) y mutación (pm), y tamaño de la población (pb) (Gómez y Cantor, 2008).

El problema de encontrar los valores óptimos de los parámetros de control ha sido objeto de estudio e investigación tanto teórica como experimentalmente (Back y Schwefel, 1993), (Gómez y Cantor, 2008). Desde el punto de vista teórico, una parametrización óptima universal no existe, ya que los valores óptimos dependen en gran parte del problema de optimización para el cual el algoritmo es aplicado (Hart y Bellew, 1991).

En el caso de los algoritmos genéticos (AG) (De Jong, 1975), en la práctica generalmente se omite el estudio minucioso del impacto de los operadores y se han adoptado como estándares parametrizaciones que han mostrado experimentalmente que funcionan para la mayoría de los problemas.

Si se quiere ser preciso, el desarrollador de un programa evolutivo (PE) debe realizar la comprobación exhaustiva considerando, con base en su experiencia, los mejores operadores y parámetros de control que garanticen la convergencia del programa y arrojen los mejores resultados.

Este proceso se realiza combinando los valores de los parámetros y operadores de manera ordenada y sistemática dentro de un rango de estudio (De Jong, 1975). Adicionalmente, cada combinación debe ser probada, por lo menos, unas 10 veces para tener resultados estadísticos relativamente válidos, dada la inicialización aleatoria de la población. Resulta imprescindible automatizar esta labor, ya que realizarla de forma manual requiere una inversión considerable de tiempo y depende en gran parte de la experiencia del desarrollador.

Se propone como solución, una herramienta enfocada en la entonación de los tres parámetros que normalmente rigen la eficiencia y eficacia de un Algoritmo Evolutivo (Candidato, 2002): la probabilidad de cruce (pc), la probabilidad de mutación (pm) y el tamaño de población (pob); así como también en la selección de los operadores de cruce (oc) y mutación (om) por su elevado impacto en la dinámica evolutiva.

La propuesta se centra en un enfoque de meta-evolución donde el algoritmo genético del nivel superior tiene la responsabilidad de colocar cualquier combinación de valores de los parámetros en los rangos válidos, sin depender de un paso particular, para optimizar los resultados del programa evolutivo de nivel inferior.

El objetivo de la herramienta es automatizar el proceso de búsqueda con el fin de propiciar la convergencia del programa evolutivo considerado, a la vez que se reduce el tiempo de desarrollo. Es importante destacar que no se plantea la búsqueda de una

Depósito Legal: PPX200002ZU2142 / ISSN:1856-4194. Volumen 13 Edición No 1 – Enero – Junio 2014

parametrización estándar aceptable en general para cualquier AE, sino un mecanismo automático de propósito general que permita conseguir la configuración óptima para cada AE específico.

Este esquema de meta-evolución es implementado mediante una arquitectura paralela maestro-esclavo utilizando el lenguaje MPICH2, con el objetivo de mejorar el tiempo de respuesta de la aplicación ya que las distintas instancias (con diferentes combinaciones) de un mismo programa evolutivo representan procesos independientes que no se comunican entre sí, presentándose el caso ideal de paralelismo. En este artículo se presenta la concepción, desarrollo y prueba de tal herramienta.

El resto de este artículo se estructura como sigue: la segunda sección describe los

FUNDAMENTOS **TEÓRICOS** de la investigación, la tercera presenta los

TRABAJOS **RELACIONADOS**; en la cuarta sección se realiza la

DESCRIPCIÓN DE LA **HERRAMIENTA** desarrollada; en la quinta sección se concentran en los

RESULTADOS **EXPERIMENTALES** y finalmente se resumen las Fuente: elaboración propia.

CONCLUSIONESresultantes.

FUNDAMENTOS TEÓRICOS

La Computación Evolutiva es un conjunto de algoritmos estocásticos de búsqueda basados en abstracciones del proceso de evolución de Darwin (Hernández, Ramírez y Ferri, 2008). En esta área se han propuesto distintos modelos computacionales denominados algoritmos evolutivos cuyo propósito es guiar una búsqueda estocástica, haciendo evolucionar un conjunto de estructuras (representativas de la solución al problema) y seleccionando de modo iterativo las más adecuadas.

La mayoría de los algoritmos evolutivos son bastante simples en su concepción, pero suficientemente complejos en su dinámica, como para proporcionar mecanismos de búsqueda robustos y potentes. El ámbito de aplicación abarca diversos campos del quehacer humano.

La población de individuos de un programa evolutivo, representa un conjunto de candidatos a soluciones de un problema. En cada iteración del algoritmo, esta población es sometida a un proceso de selección para determinar los progenitores de la próxima generación, estando favorecidos los individuos más aptos de acuerdo a una medida de adaptación.

Para generar nuevos individuos que constituyen la población hija, se aplican estocásticamente los operadores genéticos (cruce y mutación) sobre los progenitores. Si la población generada contiene los individuos deseados, se termina el proceso iterativo de

Depósito Legal: PPX200002ZU2142 / ISSN:1856-4194. Volumen 13 Edición No 1 – Enero – Junio 2014

evolución. También puede finalizar por una cota máxima de iteraciones establecidas.

Los parámetros externos de funcionamiento o parámetros de control, guían la dinámica evolutiva del algoritmo. Los más usados por su elevado impacto son: el tamaño de población (pob), la probabilidad de cruce (pc), la probabilidad de mutación (pm) y el número de iteraciones.

Los tipos de algoritmos evolutivos existentes son: Estrategias de Evolución (Schwefel, 1995), Programación Evolutiva (Fogel, 1988), Algoritmos Genéticos (Goldberg, 1989), (Holland, 1975) y Programación Genética (Koza, 1992). En este artículo se describen los algoritmos genéticos y los programas evolutivos por ser la base del trabajo reportado.

A pesar de que los algoritmos evolutivos son eficientes, algunos problemas complejos requieren de gran cantidad de tiempo de procesamiento para alcanzar una solución satisfactoria. Al respecto, las configuraciones paralelas permiten reducir el tiempo de respuesta, además, en muchas ocasiones obtienen mejores resultados que los algoritmos seriales.

ALGORITMOS GENÉTICOS Y PROGRAMAS EVOLUTIVOS

En un algoritmo genético, los individuos son representados por cadenas de bits, llamadas cromosomas. Cada cromosoma consta de un número determinado de genes. Cada gen está compuesto de uno o varios bits, de acuerdo a la naturaleza del problema. La estructura de un individuo se deriva de la representación genética conformada por cada uno de sus genes.

Al contenido de un individuo codificado en cadena binaria se le denomina genotipo y al valor específico del espacio de soluciones asociado a esa estructura se le denomina fenotipo. La población se inicializa de manera aleatoria. El método de selección permite determinar los individuos a someterse al proceso reproductivo, el cual es proporcional a la función de adaptación, utilizando algún mecanismo aleatorio como sorteo, rueda de la ruleta o torneo.

El operador de cruce realiza un intercambio estructurado de información (segmentos de bits), labor primordial para la reproducción. Actúa sobre parejas de padres y normalmente origina otro par de individuos que combinan características de sus progenitores.

Los tipos de cruce más comunes son: el cruce de un punto, el cruce de dos puntos y el cruce uniforme. El operador de mutación altera la información genética de un individuo realizando alguna pequeña modificación sobre sus genes, lo cual permite diversificar el espacio de búsqueda.

Para obtener la población hija se pueden seguir los siguientes criterios: reemplazo inmediato, donde todos los descendientes sustituyen a la población progenitora; reemplazo con factor de llenado, donde los descendientes sustituyen a los miembros de la población más similares a estos; reemplazo por inserción, donde se sustituyen ciertos

Depósito Legal: PPX200002ZU2142 / ISSN:1856-4194. Volumen 13 Edición No 1 – Enero – Junio 2014

miembros de la población (ejemplo, los peores por sus descendientes); y el reemplazo por inclusión, donde se agrupan los descendientes con los progenitores en una sola población y de ella se toman los mejores miembros.

Un criterio importante que se toma en cuenta al diseñar un algoritmo genético es el criterio de parada, que permite concretar las condiciones que indican que el algoritmo genético ha encontrado una solución aceptable, o en su defecto, ha fracasado en la búsqueda por lo que no tiene sentido continuar.

Los programas evolutivos (Michalewicz, 1992) pueden definirse como algoritmos genéticos especializados (Djordjalian, s.f) que incorporan conocimiento sobre el dominio del problema a estudiar, y donde generalmente son necesarias representaciones genéticas no binarias.

Estos surgieron al resolver problemas de optimización paramétrica que requerían alta precisión; por lo que la representación binaria obligaba a utilizar individuos de tamaño muy grande, incrementando fuertemente los requerimientos de cálculo y almacenamiento. En tales casos, los resultados mejoran significativamente (Michalewicz, 1992) usando vectores de números reales donde cada componente representa un gen.

En este tipo de programas, el operador de mutación consiste en alterar levemente el valor de una de las componentes del individuo y el operador de cruce produce dos individuos que de alguna manera promedian los valores de sus progenitores. Esto añade significado a los genotipos y a los operadores, lo cual incide favorablemente en la capacidad de procesar una mayor variedad de individuos útiles.

META-EVOLUCIÓN

Entre las técnicas más avanzadas de la Computación Evolutiva están los enfoques de meta-evolución (Back, 1994), (Freisleben, 1997), que desarrollan un esquema de dos niveles para resolver un problema. El nivel superior (nivel meta) contiene un programa evolutivo que optimiza una población de algoritmos evolutivos ubicados en el nivel inferior (nivel base).

Esta población de individuos representa instancias de un programa evolutivo que pretende solucionar un problema específico. Cada uno de los algoritmos de nivel inferior se ejecuta en forma independiente para producir una solución del problema particular. El valor de adaptación de esta solución es considerado en la operación del algoritmo de nivel superior. El algoritmo de nivel inferior con mayor valor de adaptación es considerado como el mejor programa evolutivo para el problema específico. Los números de generaciones creadas en los dos niveles son independientes uno del otro.

A continuación, se presenta la definición formal de meta-evolución según (Bäck, Fogel, y Michalewicz, 1997):

Siendo B el problema del nivel base, I_B su espacio de solución, $x = (x_1, \ldots, x_m) \in I_B$ un miembro del espacio de soluciones posibles, y $F_B: I_B \rightarrow \mathbb{R}$ la función objetivo a ser

Depósito Legal: PPX200002ZU2142 / ISSN:1856-4194. Volumen 13 Edición No 1 – Enero – Junio 2014

optimizada (sin perder generalidad se dice que la F_B debe ser maximizada, es decir, buscar por un x^* tal que $F_B(x^*) \geq F_B(x)\ \forall x \in I_B$).

Además se establece a $EA_{[V,F,I]}: V \rightarrow I$ como un algoritmo evolutivo genérico parametrizado por el espacio V de vectores que representa todas las posibles combinaciones de variantes de operadores y parámetros de control, una función de adaptación F, y un espacio I de individuos que para cada vector $v \in V$ retorna un individuo $x_v = EA_{[V,F,U]}(v) \in I$ con la mejor adaptación posible.

Basándose en esta definición genética, se considera un algoritmo evolutivo $EA_{[V_B, F_B, I_B]}: V_B \rightarrow I_B$ para el problema B. Para encontrar una buena solución al problema B, el objetivo es encontrar una configuración de parámetros $v_B^* \in V_B$ tal que:

$$F_B(\, EA_{[V_B,F_B,I_B]}(v_B^*)\,) \geq F_B(\, EA_{[V_B,F_B,I_B]}(v_B)\,)\ \forall v_B \in V_B \qquad (1)$$

Así, la búsqueda para una buena solución del problema del nivel base B se reduce a la maximización de la función objetivo $F_M: V_B \rightarrow \mathbb{R}$ del problema del nivel meta M.

$$F_M(v_B) = F_B(\, EA_{[V_B,F_B,I_B]}(v_B)\,) \qquad (2)$$

La definición de la función objetivo permite tratar la búsqueda del mejor algoritmo evolutivo para el problema B como un problema de optimización que puede resolverse por un algoritmo de meta-evolución $EA_{[V_M,F_M,V_B]}: V_M \rightarrow V_B$ donde V_M es el espacio de todas las posibles configuraciones de operadores y parámetros de control para el problema M del nivel meta. El objetivo del algoritmo de meta-evolución es encontrar óptimos valores para los parámetros:

$$v_B^* = EA_{[V_M,F_M,V_B]}(v_M) \qquad (3)$$

Donde, v_M es la configuración de parámetros y operadores utilizados en el algoritmo de la meta-evolución. La meta-evolución usualmente requiere una gran cantidad de cómputo, sin embargo, se pueden desarrollar implementaciones paralelas que se ejecuten en un tiempo razonable.

CONFIGURACIÓN PARALELA MAESTRO-ESCLAVO

Con el fin de mejorar el rendimiento y calidad de búsqueda de un algoritmo genético, investigadores en el área han propuesto técnicas de procesamiento de high performance (procesamiento paralelo y/o distribuido). La paralelización de un AG permite afrontar la lentitud de convergencia para problemas de poblaciones numerosas y algorítmicamente explotar el paralelismo intrínseco del mecanismo evolutivo (Nesmachnow, 2002).

En los AG se observa que los operadores evolutivos son computacionalmente simples con poco consumo de tiempo, mientras que el cálculo de la adaptación de los individuos sometidos al proceso de evaluación se caracteriza por consumir mayor tiempo y esfuerzo computacional (Herrera, Lozano y Verdegay, 1994). Debido a esto, se suele paralelizar el

Depósito Legal: PPX200002ZU2142 / ISSN:1856-4194. Volumen 13 Edición No 1 – Enero – Junio 2014

AG a la altura del proceso de evaluación para así distribuir la carga de procesamiento (Nesmachnow, 2002).

Se han reconocido dos modelos esenciales en el área, que han evolucionado y se han diversificado generando múltiples modelos de AG paralelo:

Modelos de poblaciones múltiples: es un enfoque orientado a la distribución de datos originando sub-poblaciones semi-independientes para resolver el mismo problema. De esta manera se analizan concurrentemente diferentes secciones del espacio de búsqueda. Los modelos de poblaciones múltiples no tienen el mismo comportamiento de un AG serial. Algunos modelos originados de este enfoque son el modelo de islas (introducción de migraciones) y el modelo de vecindad.

Modelo maestro esclavo: llamado modelo de paralelismo global, es una paralelización que, manteniendo una única población, se encarga de distribuir funcionalmente la carga computacional de las etapas del proceso de evolución, generalmente concentrada en la etapa de evaluación (Herrera, Lozano y Verdegay, 1994). Se diferencian dos modos de implementaciones, sincrónico (analizado teóricamente por De Jong en 1975) y asincrónico.

En el enfoque maestro-esclavo, un proceso maestro controla procesos esclavos con las instancias de algún programa. El maestro determina en qué momento se ejecutan los esclavos y al terminar, estos retornan datos para que el maestro realice nuevas operaciones. La meta-evolución aprovecha esta configuración paralela de manera que el maestro implementa el algoritmo genético de nivel superior, el cual optimiza instancias de un programa evolutivo, ubicadas en el nivel inferior.

TRABAJOS RELACIONADOS

Previamente, se han hecho experimentos de meta-evolución (Back, 1994; Grefenstette, 1986; De Jong, 1975; Jaramillo, 2007) para obtener valores óptimos de parámetros de algoritmos genéticos. En el ámbito de la selección de operadores, una experiencia previa (Mernik, Crepinsek y Zumer, 2000) usó meta-evolución con algoritmos genéticos para probar cuál operador de cruce es "mejor" al abordar el TSP (Travelling Salesman Problem).

Recientemente, Rivas y otros (2012) reportó experiencias iniciales aplicando el modelo meta evolutivo al asunto de la entonación de parámetros. Entre sus recomendaciones estuvo considerar de forma simultánea la asignación de parámetros y operadores de programas evolutivos, destacando no haber encontrado experiencias al respecto.

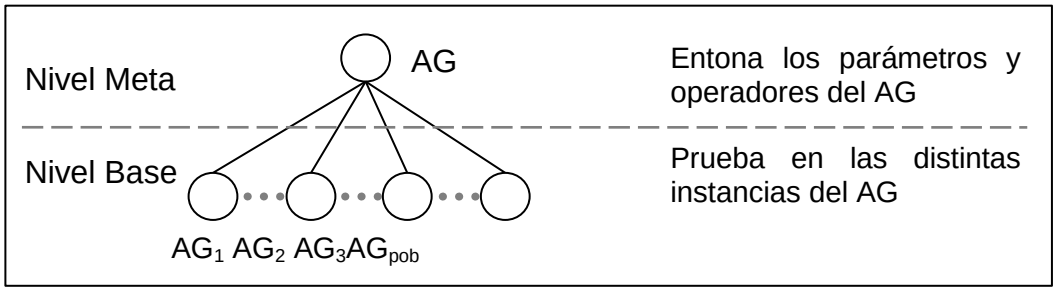
DESCRIPCIÓN DE LA HERRAMIENTA

El modelo meta-evolutivo propuesto en la presente investigación se puede observar en la Figura 1. El corazón de la herramienta lo constituye un algoritmo genético de nivel superior que controla la meta-evolución. Este fue implementado utilizando lenguaje c ansi y se comunica directamente con el programa evolutivo de nivel inferior, según este

Depósito Legal: PPX200002ZU2142 / ISSN:1856-4194. Volumen 13 Edición No 1 – Enero – Junio 2014

satisfaga las interfaces establecidas y se aloje en la ubicación correspondiente.

Figura 1. Modelo de Meta Evolución propuesto



Fuente: elaboración propia.

ARQUITECTURA DEL ALGORITMO GENÉTICO DEL NIVEL META

Corresponde a la propuesta de Algoritmo Genético Simple (AGS) (Goldberg, 1989). Esta considera una implementación básica utilizando una representación binaria para una definir una población de tamaño constante enfocada a un modelo de sustitución total, que es ejecutada con tres parámetros de entrada y aplicando tres operadores de evolución (Rodríguez y Díaz, 2009). Dicha propuesta resulta conveniente puesto que contempla la configuración completa del AGS, además de encontrarse respaldados por los estudios de Goldberg y los de De Jong.

Los parámetros de control (pc, pm, pob), el porcentaje de reemplazo, la semilla de inicialización y el número de iteraciones para este algoritmo se reciben como entrada del AG META. Los parámetros y los operadores de este nivel permanecen fijos durante la ejecución del algoritmo, siendo independientes del programa evolutivo de nivel inferior. A continuación, se presenta el resto de los detalles de diseño del algoritmo del nivel META.

ESTRUCTURA DEL CROMOSOMA

Cada individuo del algoritmo genético de nivel superior es caracterizado por un cromosoma que codifica los parámetros de control (pob, pc, pm) y los operadores genéticos (oc, om) para el programa de nivel inferior.

El cromosoma está compuesto de 32 bits, divididos en grupos que representan cada parámetro u operador a asignar, la representación genética del cromosoma se muestra en la Tabla 1.

Tabla 1. Representación del genotipo del cromosoma.

Variable	Pob	PC	PM	OC	OM	TOTAL

Depósito Legal: PPX200002ZU2142 / ISSN:1856-4194. Volumen 13 Edición No 1 – Enero – Junio 2014

# bits	8	7	10	4	3	32
# elementos	256	128	1024	16	8	4 294 967 296 combinaciones posibles
Rango	$(0-255)$	$(0-127)$	$(0-1023)$	$(0-15)$	$(0-7)$	unsgined int

Fuente: elaboración propia.

El primer grupo de ocho bits contiene un entero correspondiente al tamaño de la población, el siguiente grupo de siete bits contiene un valor real correspondiente a la probabilidad de cruce, luego un grupo de diez bits contiene el valor real de la probabilidad de mutación.

A continuación, un grupo de 4 bits contiene el operador de cruce, un valor entero entre 0 y 15; y los últimos tres bits contienen el operador de mutación, un entero entre 0 y 7. Los detalles del fenotipo asociado a cada cromosoma se puede observar en la Tabla 2.

Tabla 2. Representación del fenotipo del cromosoma.

Variable	Pob	PC	PM	OC	OM
Precisión	–	2	3	–	–
Valor Posible	$(10, 250)$	$(0.01, 0.99)$	$(0.001, 0.999)$	$(1, \text{NUM\_OC})$	$(1, \text{NUM\_OM})$
# elementos	240	99	999	NUM_OC	NUM_OM

Fuente: elaboración propia.

Donde NUM_OC y NUM_OM dependen de la cantidad de operadores de cruce y mutación compatibles con el tipo de representación cromosómica específico del programa evolutivo.

INICIALIZACIÓN DE LA POBLACIÓN

Se utiliza una misma semilla para todos los individuos del algoritmo genético superior que es pasada a todas las instancias del programa evolutivo, de manera que corran en condiciones de igualdad todos los programas del nivel base. Dicha semilla puede ser suministrada por el usuario o generada aleatoriamente por la herramienta.

MECANISMO DE SELECCIÓN

El mecanismo utilizado es el de la rueda de la ruleta, basado en una selección aleatoria proporcional a la adaptación de cada individuo (Caballero y otros, 2002).

OPERADORES DE CRUCE Y MUTACIÓN

De acuerdo al modelo AGS, se implementó el cruce de un punto y la mutación del bit.

FUNCIÓN DE ADAPTACIÓN

Dada la elevada relación entre la función de adaptación del nivel meta y la del nivel base, se decidió que en el nivel meta se considerara como adaptación el valor exacto que

Depósito Legal: PPX200002ZU2142 / ISSN:1856-4194. Volumen 13 Edición No 1 – Enero – Junio 2014

la define en el nivel base. Queda a criterio del usuario decidir si retorna al AG Meta: la mejor adaptación de la última generación, la mejor adaptación de cualquier generación, la mejor adaptación promedio de la última generación o la mejor adaptación promedio entre todas las generaciones.

Al respecto, se contemplaron los diferentes escenarios posibles, abarcando problemas de optimización numérica o combinatoria en los casos de maximización y minimización. Además se considera si las adaptaciones son siempre positivas, siempre negativas o una mezcla de ambas. En la Tabla**¡Error! No se encuentra el origen de la referencia.** 3 se presenta el resumen de la arquitectura del AG del nivel meta.

Tabla 3.Configuración del AGS del Nivel Meta.

Aspecto	Valor
Parámetros de Control	n = 100, pm =0.002 y pc = 0.8
Representación cromosómica	Cadena Binaria
Composición del gen	Un bit, o un conjunto de bits.
Población	De tamaño n constante
Inicialización	Aleatoria
Selección	Rueda de la ruleta
Operador de cruce	Cruce un punto
Operador de mutación	Mutación del bit
Modelo de sustitución de la población	Total
Función de adaptación	Según se defina en el AGbase

Fuente: elaboración propia.

ARQUITECTURA DEL PROGRAMA EVOLUTIVO DEL NIVEL BASE

La configuración de los programas evolutivos del nivel inferior obedece estrictamente a las decisiones de sus diseñadores, y la herramienta aquí presentada fue concebida para manejar cualquiera de las particularidades al respecto. Se definieron plantillas que permiten al usuario incorporar sus propios operadores a fin de que se consideren dentro del proceso de asignación.

Adicionalmente, a fin de enriquecer y facilitar la experiencia de selección de operadores, la aplicación contiene un conjunto de operadores genéticos pre-implementados que pueden ser utilizados por el programa evolutivo del nivel base, si el usuario lo desea. En este caso deben tomarse en cuenta ciertas restricciones que se detallan a continuación.

CROMOSOMAS DEL NIVEL BASE

El diseño es compatible con cromosomas de representación binaria (longitud 8, 16 y 32 bits) y representación tipo cadena de elementos, entre ellos números enteros (corto, largo,con signo, sin signo), números de punto flotante (sencillo, con doble precisión y con doble precisión largo) o letras, sumando en total 11 tipos de datos compatibles.

OPERADORES DE CRUCE Y MUTACIÓN

Depósito Legal: PPX200002ZU2142 / ISSN:1856-4194. Volumen 13 Edición No 1 – Enero – Junio 2014

Para determinar cuáles operadores serían incorporados en la aplicación se estudió la riqueza biológica, la generalidad (según el tipo de dato a que apliquen) y el principio de funcionamiento de cada uno. La aplicación gestiona de forma automática la compatibilidad entre los operadores y el tipo base de los cromosomas; sea sobre problemas de optimización numérica o combinatoria en cualquiera de sus variantes.

Para cromosomas de representación binaria se implementaron los siguientes operadores:

Operadores de Cruce : Cruce de 1 punto (SPX), Cruce de 2 puntos (DPX), Cruce de N puntos (MPX), Cruce Uniforme criterio de mascara aleatoria y un hijo (UX), Cruce Uniforme Media(HUX), Cruce Aritmético con la compuerta lógica AND (AX_AND), Cruce Uniforme con criterio de mascara aleatoria y dos hijos (UX_2), Cruce Uniforme con criterio de probabilidad 0.5 y un hijo (UX_P1), Cruce Uniforme con criterio de probabilidad de y dos hijos (UX_P2) y Cruce Aritmético con la compuerta lógica XOR (AX_XOR) (Asensio-Cuesta, Diego y Alcaide-Marzal, 2009; Orozco, 2004).

Operadores de Mutación: Mutación del bit (SMP), Mutación Inversión (IM) y Mutación del Orden (OM) (Caballero et al, 2002).

Para el modelo basado en cadenas de elementos, los operadores implementados pueden considerar restricciones sobre la repetición o no de los elementos que conforman las cadenas; ellos son:

Operadores de Cruce: Cruce por emparejamiento parcial (PMX), Cruce basado en ciclos (CX), Cruce basado en orden (OX).

Operadores de Mutación: Mutación basada en cambios (EM), Mutación basada en inserción (ISM).

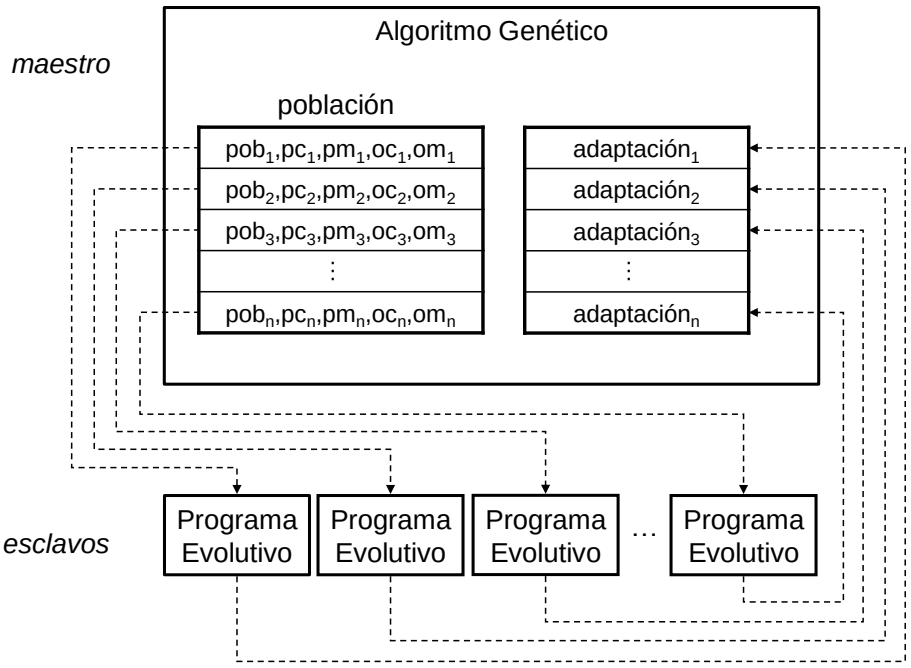
PARALELISMO DE LA HERRAMIENTA

La configuración paralela utilizada en la herramienta es la conocida como modelo maestro-esclavo. El proceso maestro ejecuta el algoritmo genético de nivel superior de la meta-evolución; mientras que los procesos esclavos son instancias del programa evolutivo, correspondientes a los diferentes individuos de la generación actual del algoritmo genético de nivel superior. Esto se muestra gráficamente en la Figura 2.

El maestro es síncrono, pues espera a que todos los individuos de la población se evalúen antes de seguir a la próxima generación. La comunicación de datos entre el maestro y los esclavos es a través de las diversas operaciones de envío y recepción del lenguaje MPICH2.

Depósito Legal: PPX200002ZU2142 / ISSN:1856-4194. Volumen 13 Edición No 1 – Enero – Junio 2014

Figura 2. Diagrama de Configuración Paralela



Fuente: elaboración propia.

En lugar de utilizar la granularidad fina, propia en AGS, el Algoritmo Genético meta se codificó utilizando un enfoque a grano grueso. De esta manera, se economizan recursos en las llamadas a la funciones de cruce y mutación del (aplicando los procesos por lotes).

El proceso maestro realiza todas las operaciones del algoritmo genético menos la función de evaluación, la cual se realiza en los procesos esclavos de forma paralela ejecutando de manera independiente las instancias del programa base. En la Figura 3 se muestra el algoritmo del proceso maestro, allí se presenta el módulo evaluar_población que levanta un proceso para cada instancia del programa evolutivo, correspondiente a cada individuo de la población. Los demás pasos del algoritmo se dan por sobrentendidos.

Depósito Legal: PPX200002ZU2142 / ISSN:1856-4194. Volumen 13 Edición No 1 – Enero – Junio 2014

Figura 3. Pseudo-Código del proceso maestro

```
// Maestro: Ejecuta Meta
        inicio
                Inicializa Población del nivel meta.
                Evaluar_Población (en Esclavos)
                Ciclo Proceso Evolutivo (Generaciones)
                        Seleccionar Padres
                        Aplicar Operadores Cruce /      Mutación
                        Reemplazar -> Nueva población
                        Envío de mensajes
                        Evaluar_Población (en Esclavos)
                        Recepción de respuesta
                Fin ciclo
        fin
// Esclavos: Ejecuta Evaluar_Población llamando al prog. evolutivo base
        inicio
                Repetir mientras que esté el ciclo evolutivo
                        Esperar por recibir un mensaje
                        Calcular adaptación llamando a Pev_base
                        Enviar respuesta a Master
                Fin repetir
        fin
```

Fuente: elaboración propia.

REPORTE DE RESULTADOS

El programa genera dos archivos de salida, el primero contiene los detalles de configuración de los algoritmos del nivel meta y base; y el segundo los detalles resultantes del proceso de entonación, destacando el mejor individuo, y los mejores individuos ordenados por tiempo de cómputo, adaptación y según la menor cantidad de iteraciones (los que aceleran la convergencia del nivel base). Un ejemplo del formato del archivo resultante del proceso de entonación puede observarse en la Figura 4.

Depósito Legal: PPX200002ZU2142 / ISSN:1856-4194. Volumen 13 Edición No 1 – Enero – Junio 2014

Figura 4. Estructura y contenido del archivo de resultados

Resultado: La mejor configuración de parámetros de todas las iteraciones

Iteración AG Meta	Individuo (ite , tp , pc , pm , gap)	OP. CRUCE	OP. MUTAC.	Aptitud IterRespuestaCromosoma PE Usuario	PE Usuario	AG Meta
26	(100, 248, 0.78, 0.058, 100)	MPX	INV	1.0000000…	1	1000110001…

Resumen detallado de las mejores configuraciones

Configuraciones de Parámetros Ordenadas por tiempo de cómputo

Iteración AG Meta	Individuo (ite , tp , pc , pm , gap)	OP. CRUCE	OP. MUTAC.	Aptitud IterRespuesta PE Usuario	PE Usuario	Cromosoma AG Meta
1	(100, 240, 0.90, 0.090, 100)	UX	SPM	0.99888706…	14	1101100011…
	... ...					
100	(100, 112, 0.65, 0.065, 100)	MPX	OM	0.99989861…	30	1011100111…

…

Configuraciones de ParámetroOrdenadas por mejor valor de Adaptación

…

Configuraciones de Parámetros Ordenadas por menor número de iteraciones PE Usuario

…

Fuente: elaboración propia.

RESULTADOS EXPERIMENTALES

La herramienta se montó en un clúster que consta de 16 nodos Sunfire V20z de doble procesador AMD® Opteron® 244, con 2G de memoria cada uno, y disco duro de 40 GB. Los nodos están conectados con Ethernet 1000/1000. El clúster se encuentra ubicado en la Facultad de Ciencias y Tecnología de la Universidad de Carabobo.

Por razones ajenas a los investigadores, solo dos nodos estuvieron disponibles para la ejecución de las pruebas. Utilizando MPICH2 se emuló un entorno de 4 procesadores. La arquitectura no se dedicó exclusivamente a los experimentos, pues a su vez debía prestar otros servicios a la Facultad.

Se escogieron tres programas evolutivos para probar la herramienta, considerando problemas de optimización de funciones y optimización combinatoria para diversificar el objetivo de los PE de nivel inferior, la cantidad de iteraciones fue de 100 en cada caso, dada la cantidad de pruebas a realizar y el elevado tiempo de ejecución requerido para cada una.

Se seleccionó un programa evolutivo para optimización de funciones (Opt_F) en el contexto de maximización, un programa evolutivo que resuelve el problema del agente

Depósito Legal: PPX200002ZU2142 / ISSN:1856-4194. Volumen 13 Edición No 1 – Enero – Junio 2014

viajero para 30 ciudades (TSP30) y otro para 51 ciudades (TSP51). Respecto al enfoque de adaptación respuesta para las pruebas, se decidió trabajar con la máxima "mejor adaptación de todas las generaciones" (adaptación global), para garantizar eficacia. En el diseño de la función de evaluación del nivel meta se consideró también la eficiencia, pues se toma en cuenta el menor tiempo (generación) en que fue lograda la máxima adaptación.

El objetivo primordial de los experimentos era verificar que la herramienta produjese los valores esperados en distintas situaciones, más allá del efecto de las semillas de inicialización del generador de números pseudo-aleatorios de lenguaje C bajo linux. Para tal fin, con cada programa evolutivo se realizaron once corridas. Se usaron tres variantes de semillas de inicialización para el nivel superior y para cada una de estas se probó con tres semillas distintas en el nivel inferior.

PROGRAMA OPT_F

La función a maximizar contiene varios máximos locales, lo cual podría generar confusión en el proceso de búsqueda del máximo absoluto en el intervalo de estudio. La Tabla 4 presenta los datos del problema de optimización abordado.

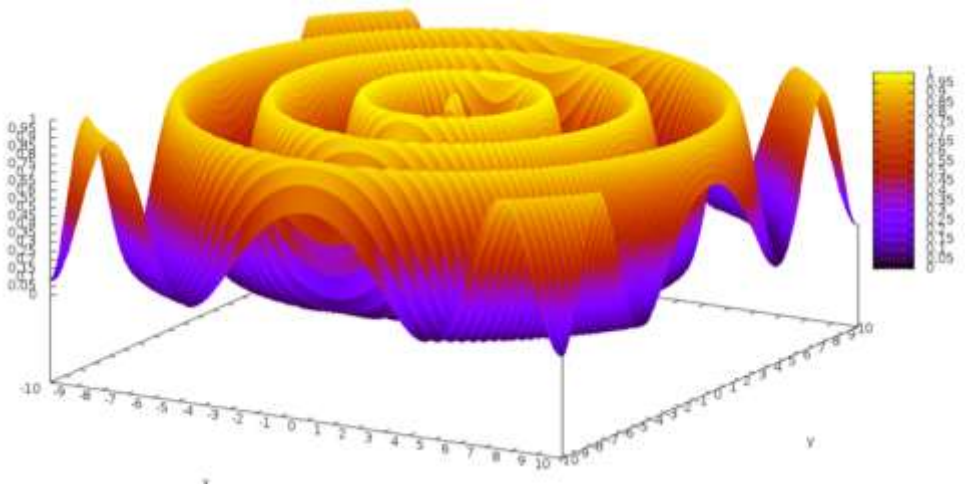
Tabla 4. Formulación del caso de prueba Opt_F

Función	$f(x,y) = 0.5 - \frac{Sen(\sqrt{(x^2+y^2)})^2 - 0.5}{1+0.001(x^2+y^2)}$
Numero de variables	2 variables (x,y)
Espacio del intervalo	$x\ (-10.0,10.0)\ y(-10.0,10.0)$
Tipo de optimización	Maximización
Valor máximo esperado	aprox.1
Implementación	Algoritmo Genético

Fuente: elaboración propia.

La Figura 5 presenta la gráfica de la Opt_F en el intervalo x ϵ [-10,10]; y ϵ [-10,10].

Figura 5. Gráfica del caso de prueba OPT_F



Depósito Legal: PPX200002ZU2142 / ISSN:1856-4194. Volumen 13 Edición No 1 – Enero – Junio 2014

Fuente: elaboración propia.

Los resultados de los 11 experimentos con el programa evolutivo Opt_F se muestran en la Tabla 5. Las columnas pob, pc, pm, se refieren a los parámetros de control seleccionados para el programa evolutivo, mientras que oc y om reportan los operadores de cruce y mutación asignados por el algoritmo genético de nivel meta. La columna Iteraciones corresponde el número de iteraciones realizadas por el programa evolutivo para conseguir la mejor adaptación.

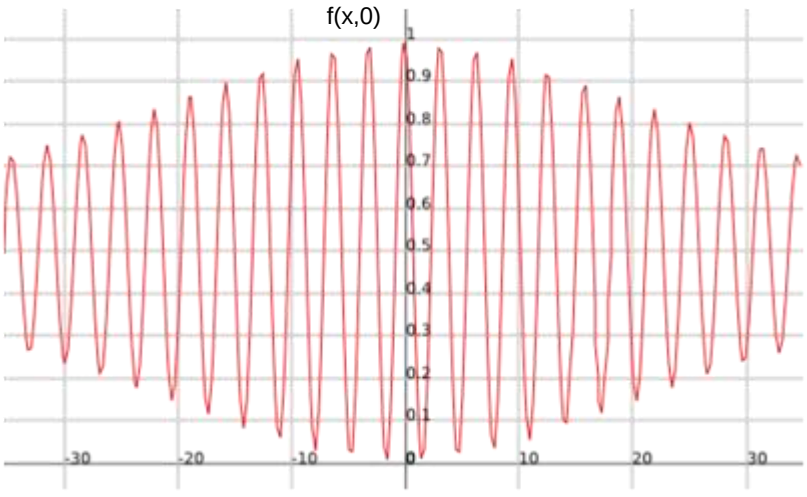
Tabla 5. Resultados experimentales del caso OPT_F

Corrida	Semilla AG	Semilla PE	pob	pc	pm	oc	om	Mejor Adaptación	Iteración Ag base
01	1083101990	1082251078	248	0.78	0.070	MPX	INV	1.000000000	1
02	1083101990	1081897119	233	0.65	0.065	MPX	OM	1.000000000	1
03	1083101990	1080213991	235	0.87	0.091	DPX	INV	1.000000000	11
04	2129989118	1082251078	187	0.84	0.095	DPX	INV	1.000000000	15
05	2129989118	1081897119	93	0.62	0.054	DPX	OM	1.000000000	16
06	2129989118	1080213991	108	0.62	0.094	MPX	INV	1.000000000	19
07	3981121976	1082251078	244	0.59	0.055	MPX	OM	1.000000000	30
08	3981121976	1081897119	127	0.96	0.059	DPX	OM	1.000000000	31
09	3981121976	1080213991	123	0.65	0.096	MPX	INV	1.000000000	36
10	3981121976	10000	116	0.65	0.085	DPX	OM	1.000000000	41
11	10000	3981121976	238	0.65	0.081	MPX	INV	1.000000000	52

Fuente: elaboración propia.

La herramienta logró alcanzar el valor esperado en todas las pruebas, en general lo consiguió en un promedio de 50 de generaciones del nivel meta. En la gráfica de la Figura 6 se observa que la mejor adaptación encontrada corresponde con el óptimo de la función.

Figura 6.Gráfica de la función OPT_F en el intervalo x ϵ [-30,30], y=0



Depósito Legal: PPX200002ZU2142 / ISSN:1856-4194. Volumen 13 Edición No 1 – Enero – Junio 2014

Fuente: elaboración propia.

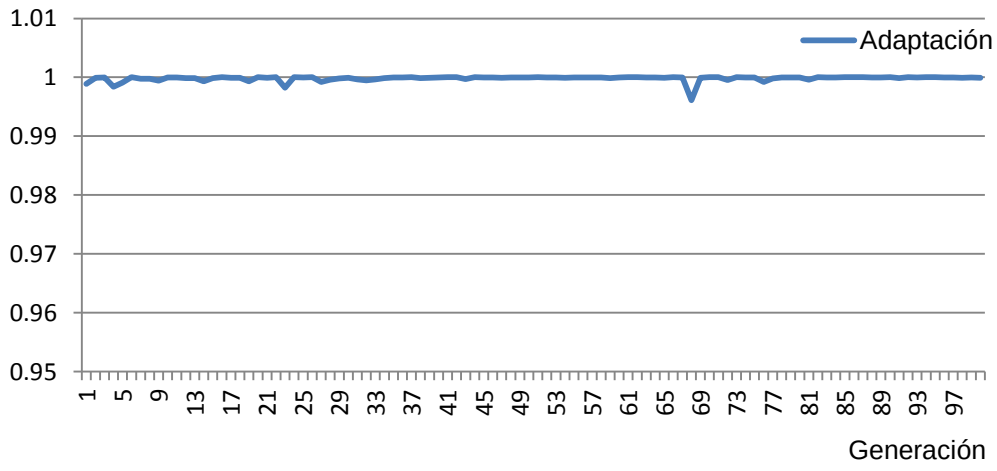
A pesar de que en todas las pruebas se alcanzó el valor esperado, se produjeron distintas combinaciones para los parámetros y operadores buscados; permitiendo alcanzar el resultado en un número de iteraciones entre 1 y 50 aproximadamente. Ante todas estas combinaciones de valores disponibles, se escogería la que alcanzara el mejor valor en el menor número de iteraciones.

Esto se debe a la influencia de las semillas utilizadas, que inicializan la población colocando los cromosomas en distintos lugares del espacio de soluciones, que favorecen o no la rápida convergencia. Por lo tanto, se puede decir que cualquiera de esas combinaciones cumple el objetivo, independientemente de la influencia de los valores de las semillas de inicialización en ambos niveles.

Para este experimento, se puede concluir que los mejores valores para los parámetros de control están en los rangos: pob $\in$ {93,..,248}, pc $\in$ [0.59, 0.96], pm $\in$ [0.054, 0.096], correspondiendo con los rangos de valores obtenidos en estudios previos (De Jong, 1975). Sin embargo, se recomienda utilizar las tuplas de valores y operadores completa, partiendo de que en esta investigación se ha medido su de forma conjunta; evitando su aplicación de manera individual.

Para el estudio de convergencia de la herramienta, las corridas arrojaron que esta converge en pocas generaciones del nivel meta (menos de 20 en promedio). En la Figura 7, se observa la evolución de los valores de adaptación para las 100 generaciones en una de las corridas del algoritmo genético de nivel superior.

Figura 7. Gráfica de convergencia del caso de prueba Opt_F



Depósito Legal: PPX200002ZU2142 / ISSN:1856-4194. Volumen 13 Edición No 1 – Enero – Junio 2014

Fuente: elaboración propia.

PROGRAMA TSP30

Esta prueba se realizó considerando un programa evolutivo para la resolución del problema del TSP utilizando como datos de entrada las ciudades del benchmark Oliver30 el cual es extraído de Oliver, Smith y Holland (1987) y cuyos datos son comparados en las investigaciones de Dorigo y Gambardella (1996). La Tabla 6 resume los detalles de este caso.

Tabla 6. Formulación del caso de prueba TSP_30

Problema	Agente Viajero, 30 ciudades.
Número de ciudades	30
Valor de la menor ruta	aprox. 423.741
Implementación	Programa Evolutivo
Longitud del cromosoma	30 genes

Fuente: elaboración propia.

El valor esperado es aproximadamente 423.741, el cual a través de las pruebas realizadas logró aproximarse. Con solo 100 iteraciones y una población de 250 individuos, se considera que aún el peor resultado constituye un valor aceptable. Las adaptaciones promedio oscilan entre 443,72839 y 500,12964, es decir, un margen de entre 20 y 80 unidades por encima del valor esperado. Los resultados obtenidos con el programa evolutivo TSP30 se resumen en la Tabla 7.

Tabla 7. Resultados experimentales del caso TSP30

Corrida	Semilla AG	Semilla PE	pob	pc	pm	oc	om	Mejor Adaptación	Iteración Ag base
01	1083101990	1082251078	196	0.91	0.815	PMX	EM	423.991027832031	42
02	1083101990	1081897119	198	0.91	0.772	PMX	ISM	425.918670654297	90
03	1083101990	1080213991	198	0.97	0.710	PMX	ISM	429.483215332031	62
04	2129989118	1082251078	198	0.91	0.678	PMX	ISM	430.511749267578	100
05	2129989118	1081897119	198	0.91	0.931	PMX	ISM	431.161956787109	91
06	2129989118	1080213991	198	0.91	0.769	PMX	ISM	429.958557128906	97
07	3981121976	1082251078	220	0.91	0.746	PMX	EM	424.292327880859	90
08	3981121976	1081897119	198	0.91	0.843	PMX	ISM	424.917633056641	93
09	3981121976	1080213991	213	0.91	0.866	PMX	EM	432.633911132812	91
10	3981121976	10000	198	0.91	0.827	PMX	ISM	425.741760253906	54
11	10000	3981121976	198	0.91	0.792	PMX	ISM	426.653991699219	93

Fuente: elaboración propia.

Todas las corridas mostraron una clara convergencia hacia el valor esperado. Por la influencia de las semillas de inicialización, en algunos casos pudo ser necesaria una mayor cantidad de generaciones del nivel meta. Los reportes muestran haber obtenido el mejor individuo alrededor de la iteración 84 (en promedio), evidenciando la necesidad de seguir explorando el proceso evolutivo entre 50 a 100 generaciones más; si se quiere mejorar la exactitud del resultado.

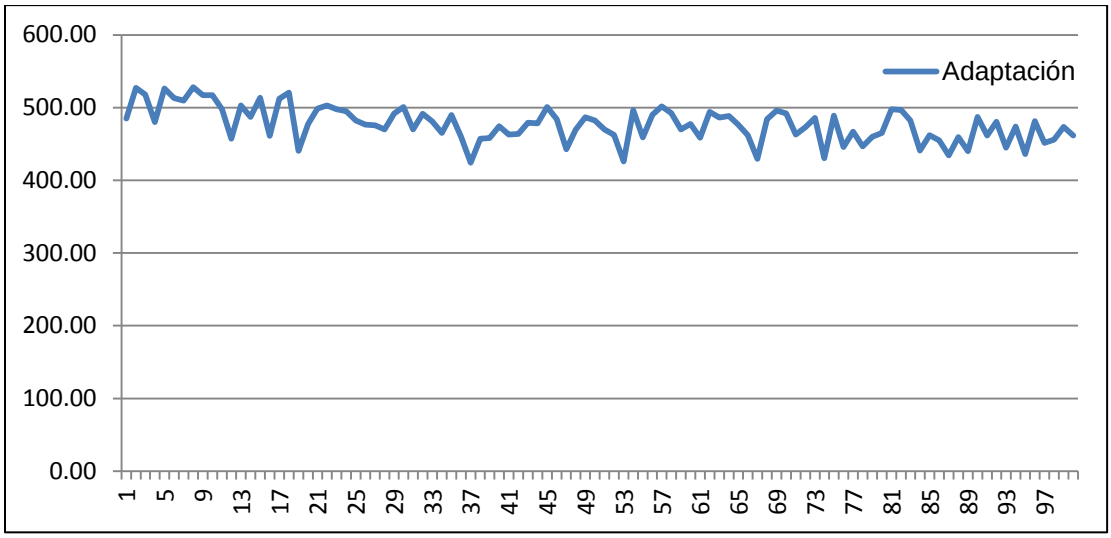
Depósito Legal: PPX200002ZU2142 / ISSN:1856-4194. Volumen 13 Edición No 1 – Enero – Junio 2014

Nótese la particularidad respecto a la probabilidad de cruce, la cual se asignó en 0.91 en 10 de los 11 casos. De igual manera, nótese como en todos los casos se seleccionó el PMX como operador de cruce. Se observa que este programa evolutivo requiere de altas probabilidades de cruce y altas probabilidades de mutación, por la complejidad del problema. A pesar de que esto puede afectar la convergencia del PE, pudiera considerarse beneficioso, conllevando a aproximar el valor esperado aún en tan poca cantidad de generaciones.

Existe una diferencia apreciable en la probabilidad de mutación (pm) encontrada por la herramienta (entre 0.67 y 0.93) y la encontrada manualmente (pm ~ 0.40), mientras que la probabilidad de cruce es similar (pc ~ 0.8). Las combinaciones de valores de los parámetros halladas por la herramienta convergen en menos iteraciones que la combinación manual (requirió entre 100 y 200 iteraciones). Con altas probabilidades de mutación, la herramienta explora más ampliamente el espacio de soluciones, desde el comienzo del proceso evolutivo.

En la Figura 8 se observa la curva de convergencia obtenida. La adaptación tiende a acercarse al valor esperado a través las generaciones.

Figura 8. Gráfica de convergencia del caso de prueba TSP30



Fuente: elaboración propia.

PROGRAMA TSP51

Programa Evolutivo para la resolución del problema del TSP utilizando los datos del benchmark Eil51.tsp; desarrollado por Eilon, Watson-Gandy y Christofides (1969); disponible en el sitio de TSPLIB: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95. La Tabla 8 resume los detalles de la solución.

Tabla 8.Formulación del caso de prueba TSP 51

Depósito Legal: PPX200002ZU2142 / ISSN:1856-4194. Volumen 13 Edición No 1 – Enero – Junio 2014

Problema	Agente Viajero, 51 ciudades.
Número de ciudades	51
Valor de la menor ruta	426
Implementación	Programa Evolutivo
Longitud del Cromosoma	51 genes

Fuente: elaboración propia.

El valor esperado de la adaptación oscila entre 426 y 430. Las pruebas realizadas no lograron alcanzar una aproximación aceptable. La mejor adaptación fue de $501{,}24$; con un aproximado de 73 unidades por encima del valor esperado.

La implementación paralela se ejecutó en tiempos superiores a las 3:00 horas para cada experimento, lo cual condicionó la realización de pruebas adicionales.

Tabla 9. Resultados Experimentales del caso TSP51.

Corrida	Semilla AG	Semilla PE	pob	pc	pm	oc	om	Mejor Adaptación	Iteración Ag base
01	1083101990	1082251078	248	0.96	0.863	CX	EM	579.524658203125	84
02	1083101990	1081897119	247	0.96	0.961	CX	EM	586.947692871094	140
03	1083101990	1080213991	188	0.96	0.89	CX	ISM	589.664123535156	88
04	2129989118	1082251078	225	0.96	0.91	CX	ISM	591.610595703125	100
05	2129989118	1081897119	248	0.94	0.83	CX	EM	593.389038085938	91
06	2129989118	1080213991	248	0.95	0.87	CX	EM	594.652404785156	93
07	3981121976	1082251078	248	0.96	0.78	CX	EM	595.068725585938	99
08	3981121976	1081897119	233	0.97	0.89	CX	ISM	598.004699707031	90
09	3981121976	1080213991	248	0.96	0.92	CX	ISM	501.247131347656	99
10	3981121976	10000	248	0.96	090	CX	EM	502.342956542969	120
11	10000	3981121976	225	0.97	0.091	CX	EM	504.462280273438	105

Fuente: elaboración propia.

Se observa nuevamente el impacto del operador de cruce manifestado en elevados valores para esta probabilidad, lo cual pudiera enriquecer la dinámica evolutiva en problemas de optimización combinatoria. De igual manera se observan elevados valores de probabilidad de mutación y del tamaño de la población, lo cual contribuye a explorar diversas áreas del espacio de soluciones y en mayor dimensión, pero incrementa considerablemente los tiempos de ejecución, principal razón por la que no se pudo experimentar con mayor cantidad de generaciones en el AG del nivel meta. Particular atención merece la preponderancia del operador CX, seleccionado como mejor operador en todos las corridas; contrastando con el operador seleccionado en la prueba TSP30.

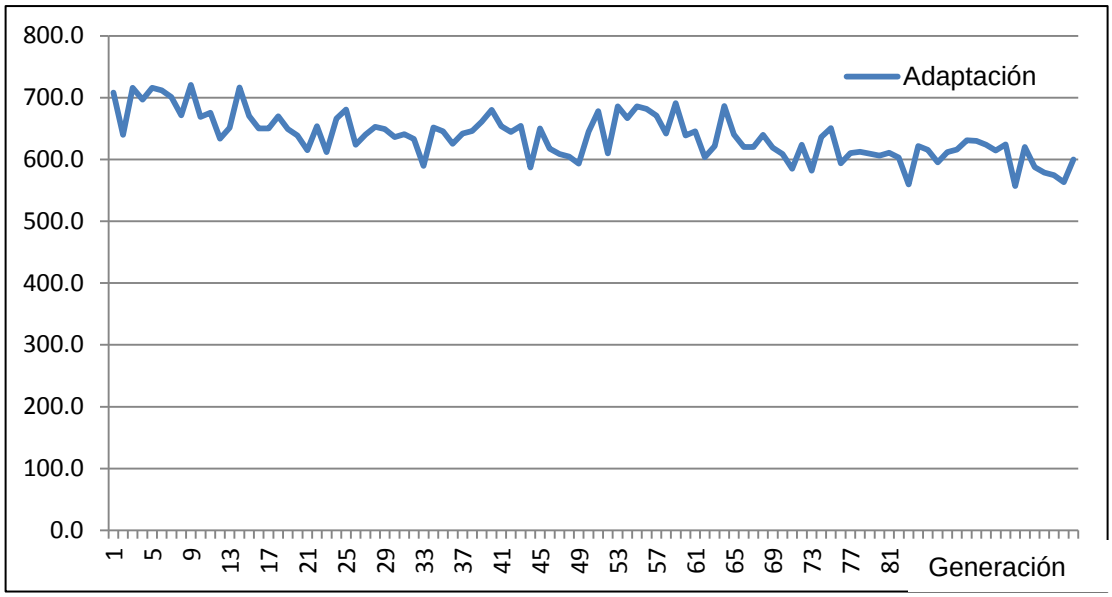
En cuanto al análisis de convergencia, se obtienen resultados similares a los obtenidos con los PEs precedentes, puesto que se observa una tendencia a converger, aunque mucho más lenta. Esto conduce a repetir este experimento en una arquitectura más poderosa y con una mayor cantidad de generaciones en el nivel meta (500 a 1000); pudiendo mejorar significativamente la convergencia.

En las pruebas manuales, se requirieron al menos 1000 generaciones para producir

Depósito Legal: PPX200002ZU2142 / ISSN:1856-4194. Volumen 13 Edición No 1 – Enero – Junio 2014

adaptaciones cercanas al valor esperado. De igual forma, se podría acelerar la convergencia considerando la incorporación de un porcentaje de reemplazo generacional en el diseño del PE base, a fin de no perder los mejores individuos entre las generaciones. La gráfica de convergencia correspondiente puede observarse en la Figura 9.

Figura 9. Gráfica de convergencia del caso de prueba TSP51



Fuente: elaboración propia.

CONCLUSIONES

La herramienta presentada automatiza el proceso de asignación de parámetros y selección de operadores de programas evolutivos, a fin de optimizar su desempeño. Su funcionamiento no depende de las particularidades de los programas evolutivos a entonar, lo cual le brinda generalidad.

Su único requisito es contar con un programa evolutivo implementado en lenguaje C ansi, que considere la interfaz comunicacional establecida: los parámetros de entrada pob, pc, pm, oc y om, retornando como salida un valor de adaptación y el número de iteración en que este se consigue.

El modelo evolutivo presentado mostró el potencial de la aplicación de algoritmos genéticos en tareas de optimización, en particular en el área de control y asignación de parámetros y operadores. En los experimentos, la herramienta consigue alcanzar su objetivo, produciendo los resultados esperados y requiriendo menor intervención y tiempo que aquellos generados manualmente por los diseñadores de los programas evolutivos probados.

Depósito Legal: PPX200002ZU2142 / ISSN:1856-4194. Volumen 13 Edición No 1 – Enero – Junio 2014

La asignación específica de los operadores evolutivos impactó significativamente la diversidad genética de las poblaciones generadas y por lo tanto la calidad de la solución generada.

El uso de una arquitectura paralela reduce significativamente el tiempo para obtener los resultados. Esta herramienta puede ser implementada en máquinas secuenciales; sin embargo, es claro que para programas evolutivos complejos, con mucha carga computacional, el tiempo de respuesta se incrementaría considerablemente.

TRABAJO FUTURO

Para trabajos futuros se contempla profundizar los experimentos para fortalecer la validez de los resultados presentados; considerando una gama de problemas más amplia y ejecutando la aplicación en una arquitectura dedicada de forma exclusiva. Ello permitiría estudiar minuciosamente el desempeño, la validez y generalidad de la herramienta, además de evidenciar las ventajas de implementar el modelo meta-evolutivo en una arquitectura paralela.

Se considera importante experimentar los efectos de la incorporación de un porcentaje de conservación en el mecanismo de reemplazo del nivel meta, evitando así la pérdida de los mejores individuos de cada generación.

Además, se recomienda incrementar el número de parámetros a ser entonados en el nivel base, entre los que se podría considerar operadores de selección y porcentaje de conservación entre generaciones, sin poner en riesgo la generalidad de la herramienta.

Finalmente, se sugiere estudiar el uso combinado de los operadores de cruce y mutación, ya que algunas investigaciones (Mernik, Crepinsek y Zumer, 2000) han sugerido como conveniente la utilización de distintos operadores en las diferentes generaciones de un mismo proceso evolutivo.

AGRADECIMIENTOS

Los autores quieren agradecer a los estudiantes de la carrera de Computación en la FaCyT - UC, que han contribuido con este proyecto, particularmente a Katherine Zaoral. Asimismo, agradecemos a Aquel que nos fortalece y conduce en esta y todas las actividades de nuestra vida: Jesucristo (1 Timoteo 1:12).

REFERENCIAS BIBLIOGRÁFICAS

Asensio-Cuesta, S.; Diego, J. y Alcaide-Marzal, J. (2009). Metodología para la generación de agendas de rotación de puestos de trabajo desde un enfoque ergonómico mediante algoritmos evolutivos. Tesis Doctoral no publicada, Universidad Politécnica de Valencia. España.

Back, T. (1994,). Parallel Optimization of Evolutionary Algorithms. InParallel Problem Solving from Nature-PPSN III: International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature, Jerusalem, Israel, Octubre.

Depósito Legal: PPX200002ZU2142 / ISSN:1856-4194. Volumen 13 Edición No 1 – Enero – Junio 2014

Back, T. y Schwefel, H. (1993). An overview of Evolutionary Algorithms for Parameter Optimization. Evolutionary Computation, Vol. 1, Núm. 1, Pp. 1-23.

Bäck, T.; Fogel, D. y Michalewicz, Z. (1997). Handbook of evolutionary computation. USA. Institute of Physics Publishing and Oxford. University Press, Bristol and New York.

Caballero, R.; Molina, J.; Luque, M.; Torrico, A. y Gomez, T. (2002). Algoritmos genéticos para la resolución de problemas de programación por metas entera. Aplicación a la Economía de la Educación. Documento en línea. Disponible en: http://www.uv.es/asepuma/X/J01C.pdf. Consulta: 12-01-2013.

Candidato Nro. 73390, (2002). Measuring the effect of population size in the efficiency of a genetic algorithm. Curso de Vida Artificial, En: Cognitive and Computing Sciences, Universidad de Sussex.

De Jong, K. (1975). An Analysis of the behaviour of a class of genetic adaptative systems. PhD thesis. Universidad de Michigan. USA.

Djordjalian, A. (s.f.). Programas de Evolución y Algoritmos Genéticos. Documento en línea. Disponible en: http://indicart.com.ar/ga/epyga.htm. Consulta: 05/01/2012.

Dorigo, M. y Gambardella, L. (1996). Ant colonies for the traveling salesman problem. Documento en línea. Disponible en: http://www.idsia.ch/~luca/acs-bio97.pdf. Consulta: 20/07/2012.

Eilon, S.; Watson-Gandy C. y Christofides, N. (1969). Distribution management: mathematical modeling and practical analysis. Operational Research Quarterly, Núm. 20, Pp. 37–53.

Fogel, D. (1988). An evolutionary approach to the travelling salesman problem. Biological Cybernetics, Vol. 60, Núm. 2, Pp. 139-144.

Freisleben, B. (1997). Handbook of Evolutionary Computation. USA. Oxford University Press.

Goldberg, D. (1989). Genetic algorithms in search, optimization and machine learning. Documento en línea. Disponible: http://download.springer.com/static/pdf/580/art%253A10.1023%252FA%253A1022602019183.pdf?auth66=1387123803_d485244feb91867e1b8197d84878d365&ext=.pdf. Consulta: 20/07/2012.

Gómez, J. y Cantor, G. (2008). Asignación de parámetros en los algoritmos evolutivos. Revista RE´TAKVN de la Facultad de Ingeniería de la Universidad del Magdalena, Vol. 1, Núm. 1, Pp. 60-67.

Grefenstette, J. (1986). Optimization of control parameters for genetic algorithms. IEEE Transactions on Systems, Man and Cybernetics, Vol. 16, Núm. 1, Pp. 122-128.

Depósito Legal: PPX200002ZU2142 / ISSN:1856-4194. Volumen 13 Edición No 1 – Enero – Junio 2014

Hart, W. y Bellew, R. (1991). Optimizing an arbitrary function is hard for the genetic algorithm. Proceedings of the 4th conference on Genetic Algorithms, Pp. 190-195.

Hernández, J.; Ramírez, M. y Ferri, C. (2008). Introducción a la minería de datos, España. Pearson Prentice Hall.

Herrera, F.; Lozano, M. y Verdegay, J. (1994). Algoritmos genéticos: fundamentos, extensiones y aplicaciones, ARBOR CLII 597, Pp. 9-40.

Holland, J. (1975). Adaptation in natural and artificial systems. USA. University of Michigan Press.

Jaramillo, J. (2007). Metodología de optimización de los parámetros de control de un algoritmo genético. Documento en línea. Disponible en: http://www.bdigital.unal.edu.co/3348/. Consulta: 20/07/2012.

Koza J. (1992). Genetic programming: on the programming of Computers by means of natural selection, MIT Press, Cambridge, Massachusetts, London.

Mernik, M.; Crepinsek, M. y Zumer, V. (2000). A meta evolutionary approach in searching of the best combination of crossover operators for the tsp. Neural Networks Nn'2000: Proceedings of the lasted International Conference. Pittsburgh, Pennsylvania. USA.

Michalewicz, Z. (1992). Genetic Algorithms + Data Structures = Evolution Programs. USA. Springer-Verlag.

Nesmachnow, S. (2002). Evolución en el diseño y la clasificación de Algoritmos Genéticos Paralelos. XXVIII Conferencia Latinoamericana de Informática. Montevideo, Uruguay.

Oliver, I.; Smith, D. y Holland, J. (1987). A study of permutation crossover operators on the travelling salesman problem. Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application. Cambridge, Reino Unido.

Orozco, M. (2004). Optimización de los parámetros de control del algoritmo genético simple usando regresión de vectores soporte. IX Simposio de Tratamiento de Señales, Imágenes y Visión Artificial. Colombia.

Rivas, J.; Perozo, F.; Rodríguez, R. y Tineo L. (2012). Una herramienta de meta-evolución paralela para la entonación de programas evolutivos. Revista Tekhné, Vol. 15, Pp. 15-19.

Rodríguez, C. y Díaz, I. (2009). Los Algoritmos Genéticos. Introducción a la inteligencia artificial. E.U.I.T. Informática de Gijón. Universidad de Oviedo, Principado de Asturias, España.

Depósito Legal: PPX200002ZU2142 / ISSN:1856-4194. Volumen 13 Edición No 1 – Enero – Junio 2014

Schwefel, H. (1995). Evolution and Optimum Seeking. Documento en línea. Disponible en: http://ls11-www.cs.uni-dortmund.de/lehre/wiley/preface.pdf. Consulta: 20/07/2012.

Depósito Legal: PPX200002ZU2142 / ISSN:1856-4194. Volumen 13 Edición No 1 – Enero – Junio 2014