ADAPTACIÓN DE LA TÉCNICA DE PARTICLE SWARM AL PROBLEMA DE SECUENCIAMIENTO DE TAREAS

RESUMEN

El problema de secuenciamiento de tareas (*Flowshop*) es un problema clásico de la programación de trabajos. La solución del modelo matemático consiste en encontrar una secuencia de tareas que emplee un tiempo mínimo de procesamiento. Este problema es considerado de difícil solución y es típico de la optimización combinatorial, por tal razón se aplicó la técnica de Partícula Swarm a la solución del problema y se validó a través de casos de prueba de la literatura especializada.

PALABRAS CLAVES: secuenciamiento de tareas, programación de trabajos, optimización combinatorial, nube de partículas.

ABSTRACT

The flow shop is a classic scheduling works problem. The mathematical model solution consists of finding a tasks sequence that uses a processing minimum time. This problem is considered of difficult solution and is typical of the combinatorial optimization, for such reason the Swarm Particle technique was applied to the problem solution and validated through specialized Literature test cases.

KEYWORDS: flow shop, works scheduling, combinatorial optimization, swarm particles.

ELIANA M TORO OCAMPO
Ingeniera Industrial, M.Sc
Profesor
Facultad de Ingeniería Industrial
Universidad Tecnológica de Pereira
elianam@utp.edu.co

YOV STEVEN RESTREPO G.
Ingeniero Electricista.
Profesor Catedrático
Programa de Ingeniería Eléctrica.
Universidad Tecnológica de Pereira
steven@ohm.utp.edu.co

MAURICIO GRANADA E.
Ingeniero Electricista, M.Sc
Profesor
Programa de Ingeniería Eléctrica
Universidad Tecnológica de Pereira
Director grupo DINOP
magra@utp.edu.co

Grupo DINOP

1. INTRODUCCIÓN

Los problemas de planificación son una prioridad dentro del plan de acción de cualquier tipo de empresa porque de ello depende, en buena parte, la rentabilidad de la misma. Dichos problemas deben ser resueltos en una amplia gama de aplicaciones tales como: programación de despacho de vuelos en los aeropuertos, programación de líneas de producción de una fábrica, programación de cirugías en un hospital, reparación de equipos o maquinaria en un taller, entre otros. Dentro de la gran variedad de problemas de planificación de recursos aparece el problema de *Flowshop*, el cual, como muchos otros en el campo de secuenciamiento de tareas, es de difícil solución y está clasificado técnicamente como de solución en un tiempo no polinomial (NP- difícil).

Un problema se considera NP-difícil cuando se demuestra que cualquier algoritmo de solución tiene un tiempo de ejecución que aumenta, en el peor de los casos, exponencialmente con el tamaño del problema [1]. Para su solución se utilizan técnicas de optimización combinatoria que permiten encontrar soluciones de buena calidad en tiempos de ejecución computacionalmente aceptables.

En este documento se implementa la técnica de optimización combinatorial conocida como nube de *partículas* (Particles Swarm Optimization - PSO) adaptada al problema de *FlowShop* y se compararan los resultados obtenidos con los que fueron encontrados aplicando el Algoritmo genético modificado de Chu-Beasley [2].

2. PROBLEMA DE FLOW SHOP

Los sistemas productivos pueden ser clasificados en tres grandes categorías: producción artesanal o por unidad (producción discreta no-repetitiva), producción mecanizada o masiva (producción discreta repetitiva), y la producción de proceso continuo**.** El problema de *FlowShop* está enmarcado en la producción mecanizada o masiva.

Cuando se pretende estudiar los problemas de secuenciamiento óptimo en líneas de producción, es posible considerar diferentes funciones objetivo, tales como: minimizar el tiempo de procesamiento de todas las tareas, minimizar la penalización por tardanzas en las fechas de entrega, minimizar el tiempo ocioso de las máquinas, entre otras. Generalmente, el problema es dividido para convertirlo en varios de un solo objetivo (optimización mono-objetivo). Sin embargo, en la literatura especializada es posible encontrar la solución al problema global considerando una optimización multiobjetivo [3],[11] cuyo propósito es encontrar una solución que minimice todos los objetivos anteriores.

Fecha de Recepción: 31 Agosto de 2006
Fecha de Aceptación: 15 Noviembre de 2006

3. DEFINICIÓN DEL PROBLEMA

El problema del *Flowshop permutacional* representa un caso particular del problema del *Flowshop scheduling* y consiste en programar, de forma óptima, un conjunto de *N* tareas que deben ser procesadas en un conjunto de *M* máquinas, considerando que todas las tareas tienen la misma secuencia de producción. El objetivo es minimizar el tiempo total requerido para terminar todas las tareas (*makespan*).

El índice *i* (*i=1,…,N*) corresponde al número de la tarea, *j* (*j=1,…M*) corresponde al número de la máquina y P_{ji} es el tiempo de procesamiento requerido por la tarea *i* en la máquina *j*. Por tal razón, un trabajo consiste de *M* operaciones y la *j-ésima* operación de cada trabajo debe ser procesada en la máquina *j*, teniendo en cuenta las siguientes restricciones:

A. *Secuencia del trabajo en el proceso*, la línea de producción se dispone de forma tal que los diferentes trabajos van en la misma secuencia de línea, (ninguno pasa a una operación posterior para luego devolverse).
B. *Reprocesos***,** **s**e considera que cada trabajo pasa Una única vez por cada operación.
C. *Ocupación***,** cada máquina sólo puede procesar una tarea por vez.
D. *Tiempos de procesamiento*, cada operación de cada trabajo tiene predeterminado su tiempo de procesamiento
E. *Tiempos de preparación***,** estos tiempos están incluidos en los tiempos de procesamiento.

*Interrupción de las operaciones***,** no deben ser interrumpidas una vez iniciadas

En este contexto se considera el trabajo i como un conjunto de operaciones, pasando por cada máquina j una sola vez:

- $i = \{o_{1i}, o_{2i}, o_{3i}, ..., o_{ji}\}$ donde o_{ji} representa la *j-ésima* operación del trabajo i y debe ser procesada en la máquina *j*;
- Por cada operación o_{ji} hay asociado un tiempo de procesamiento p_{ji}.

A continuación se ilustra un ejemplo de cómo se procede para evaluar el *makespan*. En este caso se solicita el ordenamiento de 4 trabajos que deben ejecutarse en 3 máquinas cumpliendo con los tiempos por operación presentados en la tabla 1.

P_{ij}	$i=1$	2	3	4
$j=1$	4	4	3	2
2	7	3	2	1
3	5	5	3	1

Tabla 1. Tiempos de procesamiento.

En este artículo una secuencia de trabajos es representada como X= [π_1 π_2 π_3 ...π_i ... π_N]. La figura 1 ilustra la obtención del *makespan* para la secuencia de trabajos X=[4-3-2-1]. El tiempo total de ejecución es de 24 unidades de tiempo.

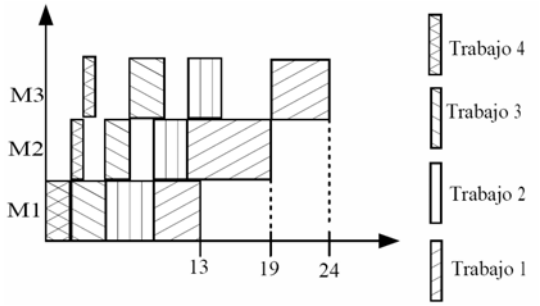

Figura 1. Obtención del *Makespan*.

El modelo matemático utilizado para la solución del problema tratado, define los siguientes parámetros:
Sea $\pi_1, \pi_2, \pi_3, ..., \pi_N$ la secuencia de trabajos a realizar. El procedimiento para el cálculo de $C(\pi_i, j)$, que representa el tiempo total de ejecución para el *i-ésimo* trabajo de la secuencia π en la máquina *j*, inicia contabilizando el tiempo que tarda el trabajo *i* en salir de la primera máquina mediante el conjunto de ecuaciones (1).

$$\begin{aligned} &C(\pi_1,1) = p_{\pi 1},1 \\ &C(\pi_2,1) = C(\pi_{2-1},1) + p_{\pi 2},1 \\ &\vdots \\ &C(\pi_i,1) = C(\pi_{i-1},1) + p_{\pi i},1, \quad con \quad i = 2,...,N \end{aligned} \tag{1}$$

Posteriormente se realiza idéntico procedimiento pero ahora en la máquina dos (conjunto de ecuaciones -2-).

$$\begin{aligned} &C(\pi_1,2) = C(\pi_i,1) + p_{\pi 1},2 \\ &C(\pi_2,2) = C(\pi_{2-1},2) + p_{\pi 2},2 \\ &\vdots \\ &C(\pi_i,2) = C(\pi_{i-1},2) + p_{\pi i},2, \quad con \quad i = 2,...,N \end{aligned} \tag{2}$$

Generalizando, el $C(\pi_i, j)$ puede ser calculado como:

$$C(\pi_i, j) = C(\pi_i, j-1) + p_{\pi i}, j, \tag{3}$$

Bajo estas especificaciones, el *makespan* ($C_{\max}$) está dado por $C(\pi_N, M)$, que corresponde a el tiempo completo del último trabajo en la última máquina [4], [5].

4. METODOLOGÍA DE SOLUCIÓN

Para la solución del problema se emplea la técnica de optimización combinatorial conocida como Nube de Partículas (Particles Swarm Optimization - PSO), basada en una población de soluciones y guiándose a través de la inteligencia colectiva de dicha población. Está técnica de optimización está inspirada en el comportamiento social

de los animales tales como una bandada de pájaros, un cardumen de peces o un enjambre de abejas. PSO ha demostrado ser eficiente en problemas multidimensionales continuos, en problemas de investigación de operaciones tales como: identificación de procesos, proyectos de controladores, localización de bancos de capacitares, entre otras [6],[7].

En la tabla 2, se presenta una serie de términos que se emplean en el algoritmo PSO tradicional.

Partícula/ Agente	Un individuo de la Nube
Localización/ posición	Coordenadas de un agente en un espacio N-dimensional que representa una solución para el problema.
Nube	Toda una colección de agentes. Una población de individuos.
Fitness	Un número que indica la calidad de una solución dada (representada por una localización en el espacio de soluciones).
pbest (Personal best)	Es la mejor localización obtenida por un determinado agente a lo largo del proceso.
gbest (Global best)	Es la mejor localización en toda la nube de partículas.
Vmax	Es la velocidad máxima permitida en una dirección dada.

Tabla 2. Términos del Algoritmo PSO.

El algoritmo de nube de Partículas parte de una población para iniciar un proceso de búsqueda, donde cada individuo se refiere a una partícula que está agrupada dentro de una nube (un pájaro, dentro de una bandada). Cada partícula dentro de la nube representa un candidato a la solución del problema de optimización. En PSO cada partícula se desplaza a través de un espacio de búsqueda multidimensional, ajustando su posición de acuerdo a su propia experiencia y la de otras partículas en su vecindario, es decir, una partícula hace uso de la mejor posición encontrada por si misma (*pbest*) y de la mejor posición de su vecindario (*gbest*), para dirigirse hacia la solución optima.

4.1 *Algoritmo básico*

A continuación se mostrar los pasos a seguir en el proceso de optimización utilizando PSO tradicional:

i) Se inicia con una población de partículas, con posiciones y velocidades en el espacio del problema n-dimensional generado de forma aleatoria.

ii) A cada partícula se le evalúa la función objetivo.

iii) Se compara el valor de la función de adaptación (*fitness)* de la partícula con el *pbest* de la partícula. Si su valor corriente es mejor que el *pbest* pasa a ser igual al valor de la *fitness* de la partícula y la localización del *pbest* pasa a ser igual a la localización actual del espacio n dimensional.

iv) Se compara el valor de la función de *fitness* con el mejor valor de adaptación de la población. Si el valor actual es mejor que el *gbest*, actualizar el valor de *gbest*

v) Modificar la velocidad de la posición de la partícula de acuerdo a las ecuaciones (2) y (3), respectivamente:

$$v_{i(t+1)} = w * v_i(t) + c_1 * ud() * [pbest(t) - x_i(t)] + c_2 * Ud() * [gbest(t) - x_i(t)] \quad (2)$$

$$x_i(t+1) = x_i(t) + \Delta t * v_i(t+1) \quad (3)$$

vi) Volver al paso *ii)* hasta que el criterio de parada sea encontrado, usualmente se define con un número máximo de iteraciones $t_{\max}$.

La nomenclatura usada es:

$x_i = [x_{i1}, x_{i2}, ..., x_{in}]^T$ guarda la posición de la i-ésima partícula.

$v_i = [v_{i1}, v_{i2}, ..., v_{in}]^T$ almacena la velocidad de la i-ésima partícula.

$pbest = [p_{i1}, p_{i2}, ..., p_{in}]^T$ representa la posición del mejor valor *fitness* de la i-ésima partícula.

w = ponderación de inercia.

c_1 , c_2 constantes positivas.

$ud(), Ud()$ son dos funciones para generar números aleatorios con distribución uniforme [0,1].

$v_{\max}$ guía la búsqueda a través del espacio de soluciones, si su valor es alto facilita la búsqueda global y si es pequeño facilita la búsqueda local.

La primera parte de la ecuación (1) es un término de momento de la partícula donde w es grado del momento; la segunda parte consiste en la parte “cognitiva” que representa el “conocimiento” independiente de la partícula y la tercera parte es la parte “social” que representa la colaboración entre las partículas.

c_1 , c_2 representan ponderaciones de la parte cognitiva y social, respectivamente e influencian cada partícula en dirección a las posiciones *pbest* y *gbest* . Estos parámetros son ajustados generalmente a tentativa y

error, al tamaño de la población o dependiendo de la naturaleza del problema

5. ADAPTACIÓN DEL ALGORITMO PSO AL FLOW SHOP

Tradicionalmente en el algoritmo de PSO cada partícula representa una solución en el parámetro del espacio. La partícula es codificada como una cadena de posiciones las cuales representan un espacio multidimensional. Todas las dimensiones típicamente son independientes unas de otras, sin embargo los cambios en la velocidad y en la partícula son efectuadas independientemente en cada dimensión, ese es uno de los aspectos relevantes de PSO. Esta característica no es aplicable a los problemas de permutación porque los elementos no son independientes unos de otros, en el desarrollo del algoritmo existe la posibilidad de que dos o más posiciones puedan obtener el mismo valor, factor que rompería con la regla de la permutación, quiere decir esto que aparecerían configuraciones infactibles, para eliminar este conflicto se debe emplear una estrategia adicional.

En PSO, para actualizar la ubicación de la partícula en el espacio multidimensional se adiciona la velocidad a la posición actual (ecuación 3). Si la velocidad es grande, la partícula puede explorar grandes áreas, por tanto la nueva velocidad en un escenario permutacional que representa la posibilidad de que la partícula cambie. Si la velocidad es grande la partícula tendrá una probabilidad alta de cambiar a una nueva secuencia de permutación. La velocidad se actualiza de la misma manera que en el algoritmo básico de PSO (ecuación 2), sin embargo la velocidad estará limitada por los valores absolutos, ellos sólo representan la diferencia entre las partículas.

El proceso de actualización de la partícula se hará de la siguiente forma: la velocidad es normalizada en un rango de 0 a 1 dividiendo todos los elementos por el máximo valor de la cadena, luego cada posición determinará si se hace o no el intercambio con una probabilidad determinada por la velocidad (ver figura 2). Seguidamente se procede a seleccionar aleatoriamente la dimensión de la partícula que debe cambiar. Dependiendo del valor de la probabilidad para esa dimensión (*vnorm*, en la figura 2 la posición señalada), se decide si se hace el cambio o no. Si el intercambio es requerido la posición será ajustada al valor en la misma posición del *pbest* o del *gbest* por intercambio de valores [8].

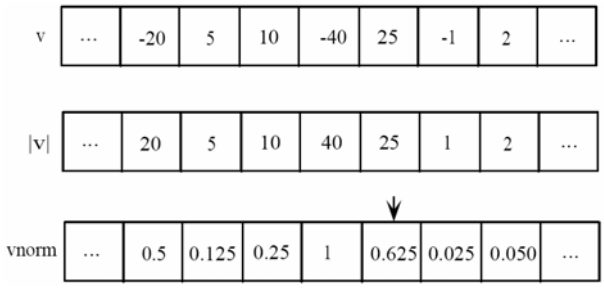

Figura 2. Normalización velocidad de la partícula.

La posibilidad de que la partícula migre hacía el *pbest* o hacia el *gbest*, es dependiente del deseo de realizar una explotación de la mejor posición encontrada por la partícula a lo largo de su historia (*pbest*) o de procurar por una exploración mas amplia de posibilidades en el espacio de soluciones dirigida por la mejor posición de la nube de partículas a lo largo de todo su desplazamiento (*gbest*). Con el objetivo de no sesgar la búsqueda, en este artículo se permite que la partícula en un momento determinado, se desplace en dirección del *pbest* o del *gbest*, de acuerdo a una competencia que realizan las dos opciones en una ruleta de probabilidades. En este artículo se le dio una opción del 60 % para que la partícula pudiera migrar en un momento determinado hacía la mejor posición toda la población alcanzada a lo largo de su evolución y de un 40 % hacía su mejor posición a lo largo de su historia. En la figura 3 se muestra cual es el proceso de actualización de la posición de la partícula, después de la normalización de la velocidad (suponiendo que la partícula migra hacía el *gbest*).

Cabe recordar que cada posición representa una secuencia de tareas (en la figura 3, las posiciones corresponden a X y a $X+v$).

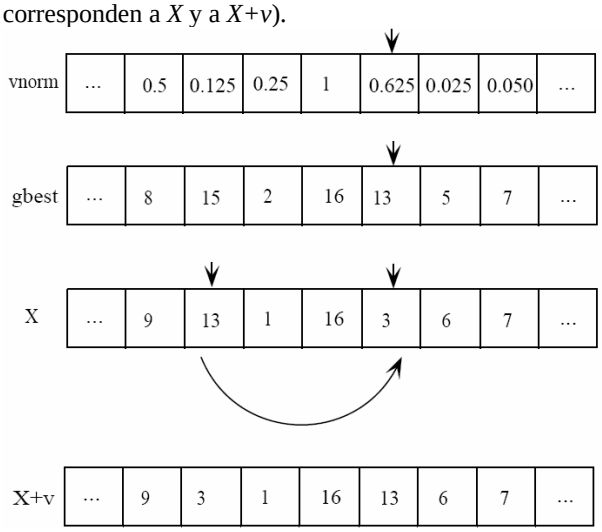

Figura 3.Proceso de actualización de la posición.

Para prevenir que las partículas se queden atrapadas en óptimos locales, durante la etapa de actualización de las posiciones se recurre a un proceso de permutación

adicional (*mutación*, intercambiar tareas en la secuencia de producción).

5.1 *Resultados*

Para la adaptación de la técnica combinatorial y de las modificaciones hechas a su algoritmo tradicional, se utilizaron tres casos de prueba Taillard [2] (ver características en la tabla 3).

Caso	# de Tareas	# de Máquinas	Cmax	
			Límite Inferior	Límite Superior
ta_20_5_01	20	5	1232	1278
ta_50_5_01	50	5	2712	2724
ta_100_5_01	100	5	5437	5493

Tabla 3. Casos de prueba.

La implementación del algoritmo de PSO adaptado al problema de *flowshop* se desarrolló en el entorno de Borland Delphi 7.

Los resultados obtenidos para cada uno de los casos de prueba se muestran en la tabla 4. Se puede notar la buena adaptación del método al problema tratado en este artículo, ya que se encontraron secuencias de trabajo para cada uno de los casos, que permitieron obtener un *makespan* (Cmax) que esta dentro del rango exigido por la literatura especializada, para considerar la secuencia encontrada de excelente calidad.

CASO	Número de Partículas	Número de generaciones necesarias para alcanzar función objetivo	Cmax
ta_20_5_01	60	39	1278
ta_50_5_01	150	210	1724
ta_100_5_01	500	709	5493

Tabla 4. Resultados.

En la figura 4, se puede observar cual fue la evolución del método adaptado de PSO, para el caso ta_100_5_01. En un principio la metodología evoluciona rápidamente tras la búsqueda de una solución de calidad aceptable, permanece en rango de 10 a 708 iteraciones, explotando las posibilidades hasta que llega a una solución de excelente calidad en la iteración 709 (Cmax=5493, presentada en la tabla 4).

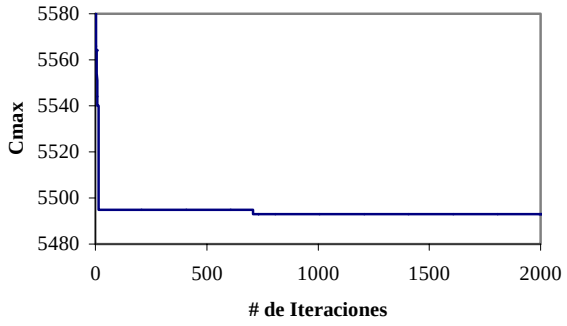

Figura 4. Evolución de la función objetivo para el caso ta_100_5_01.

6. CONCLUSIONES

La adaptación de la técnica de optimización combinatorial PSO a el problema permutacional de *flowshop*, arrojó resultados de excelente calidad, proporcionando secuencias de trabajo para cada uno de los casos de prueba utilizados, obteniéndose un *makespan* que de acuerdo a la literatura especializada está dentro de los límites permitidos (Ver tabla 3). Estos resultados son similares a los que se obtuvieron utilizando la técnica de Algoritmo Genético modificado en [2].

PSO es una de las más recientes técnicas de optimización aplicadas a la solución de problemas combinatoriales, la literatura especializada la presenta como una metodología de desempeño eficiente, afirmación que fue comprobada en el desarrollo de este proyecto, queda por explorar su efectividad en otros temas de aplicación tales como: solución del problema de asignación generalizada, asignación óptima de salones, problemas de empaquetamiento, entre otros.

La etapa de factibilización que requiere un algoritmo basado en poblaciones, como en este caso, no fue necesario implementarla debido a la modificación que se hizo al PSO básico, esta situación es muy conveniente debido a que se minimiza el tiempo computacional de ejecución.

Entre las adaptaciones que se pueden plantear para la técnica PSO hay una muy interesante que es segmentar la población y a cada segmento asignar un *gbest* local, la hipótesis es que con está modificación la búsqueda en el espacio de soluciones podría ser más intensiva.

7. AGRADECIMIENTOS

Los autores expresan su agradecimiento a la Universidad Tecnológica de Pereira por su apoyo al grupo de desarrollo en Investigación de Operaciones - DINOP

8. BIBLIOGRAFÍA

[1]GAREY M.; JOHNSON. Computer and Intractability: A guide to the theory of Np- complétense. Freeman. New York,1979.

[2]TORO, Eliana.; RESTREPO Yov; GRANADA Mauricio. Algoritmo genético modificado aplicado al problema de secuenciamiento de tareas en sistemas de producción lineal- Flowshop. Scientia et Técnica. No.30 Mayo de 2006, Universidad Tecnológica de Pereira.

[3]Cortes Rivera Daniel. Un sistema inmune artificial para resolver el problema de Job Shop Scheduling. CINVESTAV, México D.F. Marzo de 2004.

[4]R.Z. Ríos-Mercado y J.F.BArd. Secuenciando óptimamente líneas de flujo en sistemas de manufactura. Revista de Ingenierías.Universidad de Nuevo León . México Enero-marzo 2001, Vol IV, No 10.

[5]R:Z: Ríos-Mercado y J:F:Bard An enhanced TSP-based heuristic for makespan minimization in a Flowshop with setup times. Journal of Heuristics, 5(1):57-74,1999.

[6]Eberhart, R, C and Kennedy, J. A new optimizer using particle swarm theory. Proceedings of the Sixth International Symposium on Micromachine and Human Sciencie, Nagoya, Japan, pp 39-43.1995

[7]Kennedy, J and Eberhart. Particle swarm optimization. Proceedings of IEEE International Conference on Neuronal Networks, Piscataway, NJ.pp 1942-1948, 1995.

[8]Hu, Xiaohui; Rusell E.;Shi Yuhui. Swarm Intelligence for Permutation Optimization: A case study of n_Queens Problem. Proceedings of the IEE Congress on Evolutionary Computation (CEC 2002), Honolulu, Hawaii, USA 2002

[9]Shi, Y and Eberhart, R:C A modified particle swarm optimizer proceedings of the IEEE Congress Evolutionary Computation(CEC 1998). Piscataway. NJ pp 69-73, 1998.

[10]Van den Bergh,F and Engelbrecht, A P. Training product unit networks using cooperative particle swarm optimizasers. Proceedings of INNS-IEEE International Joint Conference on Neuronal Networks 2001, Washington DC, USA.2001

[11]Coello Coello C.A. and Lechuga, M.S. MOPSO: a proposal for multiple objective particle swarm optimization. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002) Honolulu, Hawaii USA. 2002.