TSP, QAP, VRPTW:  
Resolución mediante Algoritmos MOEA: SPEA, NSGA y Algoritmos ACO: M3AS, MOACS

Marcelo Ferreira, Christian Gómez, Guido Casco,

Alida Invernizzi.

Electiva III, Inteligencia Artificial, Octavo Semestre,

{jmferreira1978, cgomezpy, guiancs82, alidainvernizzi}@gmail.com

**Resumen.** El documento presentado trata a cerca de las implementaciones realizadas para resolver los problemas del TSP, QAP, VRPTW, por medio de Algoritmos Multiobjetivos Evolutivos tales como SPEA y NSGA, así como también con los Algoritmos de Colonia de Hormigas tales como M3AS y MOACS.

**Palabras Claves:** Optimización multi-objetivo, colonias de hormigas, frente pareto

1. Introducción

Este documento muestra una comparación entre diversos algoritmos que utilizan la optimización multi-objetivo, dos de ellos con enfoque evolutivo y otros dos basados en el modelo del comportamiento de las colonias de hormigas reales, denominada metaheurística ACO (*AntColony Optimización*, u optimización basada en colonia de hormigas).

El trabajo considera algoritmos propuestos recientemente como el M-MMAS[Pinto05] y el MOACS [Paciello06]…

Se realizaron pruebas con dos instancias de cada problema. Se utilizaron reconocidos problemas de prueba de optimización multi-objetivo, el *Quadratic Assignment Problem* (QAP), definido en [Knowles03], el *Traveling Salesman Problem* (TSP) [Garcia04] y el *Vehicle Routing Problem with Time Windows* (VRPTW) [Baran03]. Estos problemas son considerados clásicos en la literatura de optimización combinatoria y del tipo NP-completos.

El trabajo está organizado como sigue: en la sección 2 se explica conceptos fundamentales sobre la optimización multiobjetivo, en la sección 3 se trata la formulación matemática de la optimización multiobjetivo y una descripción de los problemas, en la sección 4 se describen los algoritmos multi-objetivos utilizados. Los resultados experimentales de la comparación se muestran en la sección 5, y finalmente en la sección 6 se presentan algunas conclusiones y trabajos futuros

1. Conceptos de la Optimización Multiobjetivo

La optimización multi-objetivo puede ser definida como el problema de encontrar un vector de variables de decisión que satisfacen restricciones y optimiza un vector de funciones cuyos elementos representan las funciones objetivo. Estas definiciones aparecen en los trabajos de [Coello99] y [Deb99].

Optimizar

|  |  |
| --- | --- |
|  |  |

Sujeto a

|  |  |
| --- | --- |
|  |  |
|  |  |

Donde es el *vector de decisiones* con los valores para las *N* variables de decisión del problema, es el *vector solución* con las evaluaciones de las *M* funciones objetivos , y las funciones y representan respectivamente las *J* restricciones de desigualdad y las *K* restricciones de igualdad sobre el espacio de las variables de decisión. Por *optimizar* se entiende la *minimización* o *maximización* de cada una de las *M* funciones objetivas.

**Definición 1**: *Dominancia de Pareto*: Sean dos soluciones . Se dice que *u* domina a *v* (denotado como ) si es mejor o igual que *v* en cada uno de los objetivos y estrictamente mejor en al menos un objetivo. Como ejemplo, en un contexto de minimización si y solo si:

**Definición 2**: *Soluciones no comparables*: Dados , si ni , se dice que son soluciones no comparables, lo que se denota como *u ~ v*.

**Definición 3**: *Conjunto Pareto*: El conjunto de todas las soluciones no dominadas en se denomina Conjunto Pareto, lo que se denota como CP. Las soluciones que pertenecen a CP se denotarán como *x\**.

**Definición 4**: La imagen del Conjunto Pareto a través de la función se denomina Frente Pareto, denotado por *Y*.

1. Descripción de los Algoritmos

En la figura 1 se muestra el pseudo-código de un algoritmo ACO multi-objetivo genérico, denominado en adelante MOACO (MultiObjective Ant Colony Optimization). El MOACS y M3AS, presentado a continuación, siguen éste pseudo-código.

|  |
| --- |
| procedure MOACO  inicializar\_parametros()  while not condicion\_parada()  generacion=generacion + 1  for ant=1 to m // m es la cantidad de hormigas  construir\_solucion()  evaluar\_solucion()  actualizar\_feromonas()  actualizar\_conjunto\_pareto()  end for  end while  end  procedure construir\_solucion  sol={Ø}  while existen\_estados\_no\_visitados()  siguiente=seleccionar\_siguiente\_estado()  sol=sol U {siguiente}  marcar\_como\_visitado(siguiente)  if(actualizacion\_paso\_a\_paso)  actualizar\_feromonas\_paso\_a\_paso()  end while  end |

Fig. Pseudo-código de un algoritmo ACO multi-objetivo genérico.

* 1. MultiObjective Ant Colony System (MOACS)

MOACS, propuesto por Barán y Schaerer en [Baran03], es una extensión del MACS-VRPTW, este último propuesto por Gambardella et al. [Gambardella99]. Fue implementado considerando dos objetivos, utiliza una matriz de feromonas y dos visibilidades, una para cada objetivo del problema. La regla de transición de estados se calcula como:

|  |  |
| --- | --- |
|  |  |

El cálculo de se realiza según:

|  |  |
| --- | --- |
|  |  |

Cada vez que una hormiga se mueve del estado *i* al estado *j*, realiza la actualización local de feromonas según la ecuación:

|  |  |
| --- | --- |
|  |  |

con , el valor inicial para las feromonas, y representa el coeficiente de evaporación y,

|  |  |
| --- | --- |
|  |  |

con donde por motivos de normalización los valores son divididos por un valor máximo definido a priori.

En el caso de encontrar una solución no dominada, se actualiza CP y se reinicializa la tabla de feromonas, considerando que la información fue aprendida por medio de soluciones dominadas. Si la solución encontrada es dominada se realiza la actualización de feromonas según la ecuación (6).

* 1. Multiobjective Max-Min Ant System (M-MMAS o M3AS)

Este algoritmo, propuesto por Pinto et al. en [Pinto05], extiende el Max-Min Ant System para resolver problemas multi-objetivos. Se utilizó inicialmente para resolver el problema de enrutamiento multicast multi-objetivo. Pinto et al. en su trabajo [Pinto05] optimizaron cuatro objetivos. Se mantiene una tabla de feromonas global que mantiene información de feromonas considerando todos los objetivos a optimizar. Una hormiga estando en el estado *i* escoge el siguiente estado a visitar, de acuerdo con la probabilidad *p* dada según la ecuación (5).

Las soluciones no dominadas actualizan la tabla de feromonas según la ecuación (6). Si , entonces , con y si , entonces , con , donde *m* es el número de hormigas, *k* representa la *k-ésima* solución y se calcula con la ecuación (7). De esta manera se impone una cota inferior y otra superior al nivel de feromonas en los arcos.

* 1. Algoritmos Evolutivos

Los Algoritmos Evolutivos (EA – *Evolutionary Algorithms*) se basan en la simulación de un proceso de evolución Darwiniano sobre una población de individuos que representan soluciones potenciales al problema que se pretende resolver [Zitzler00, Lücken03]. Este proceso evolutivo está soportado a través de la aplicación iterativa de las operaciones de selección, cruzamiento, mutación y reproducción sobre los individuos de la población.

La operación de *selección* consiste en la identificación de los individuos que serán sometidos a las operaciones de cruzamiento y mutación. La operación de *cruzamiento* consiste en la generación de nuevos individuos a partir de la combinación de dos o más individuos de la población, donde los nuevos individuos se dice que poseen características que son similares o heredadas de sus progenitores. La operación de *mutación* consiste en la modificación de la información contenida en el individuo de manera aleatoria, la cual suele realizarse de acuerdo a cierta probabilidad. Por último, la operación de *reproducción* consiste en la identificación de los individuos más aptos en la población actual que conformarán la población para la siguiente generación.

A continuación se muestra el pseudocódigo genérico de un EA:

1. t := 0;

2. Inicializar P(0);

3. Evaluar P(0);

4. DO

5. P’(t) := Cruzar(Seleccionar(P(t));

6. Mutar(P’(t));

7. Evaluar(P’(t));

8. P(t+1) := Reproducir(P(t) U P’(t));

9. t := t+1;

10. WHILE(condición\_de\_parada() ==FALSE)

11. Retornar mejor solución;

**Fig. 2** Pseudocódigo genérico de un Algoritmo Evolutivo.

Los EAs comienzan con la creación, inicialización y evaluación de una población inicial de individuos generados al azar, la cual será refinada mediante la aplicación iterativa de las operaciones de cruzamiento, mutación, selección y reproducción.

La operación de *evaluación* consiste en la obtención de un *valor de adaptabilidad* (*fitness*) para cada individuo, el cual indica que tan buena es la solución representada por el individuo para el problema que se pretende resolver.

Así, en cada generación *t* se genera una población de individuos *P’*(*t*) mediante la aplicación de la operación de cruzamiento sobre un conjunto de individuos seleccionados a partir de su valor de adaptabilidad de la población *P*(*t*). Luego, los individuos de esta población *P’*(*t*) son sometidos a la operación de mutación. Por último, mediante el proceso de reproducción se determinan los individuos de *P*(*t*) y *P’*(*t*) que conformarán la población para la siguiente generación *P*(*t+*1).

La función *condición\_de\_parada*() devuelve un valor booleano que indica la terminación de las iteraciones del EA, lo cual se da una vez que se haya encontrado una solución deseada, haya transcurrido un cierto número de generaciones o se alcance un cierto tiempo de ejecución.

Cabe hacer notar que la manera en que se aplican e implementan los operadores de selección, cruzamiento, mutación y reproducción diferencia a las distintas técnicas basadas en la teoría de EA, la cual se divide en los paradigmas descritos a continuación [Lücken03,Toscano01].

* + 1. Strength Pareto Evolutionary Algorithm

El *Strength Pareto Evolutionary Algorithm* (SPEA), propuesto por Zitzler et al. [Zitzler98], es un método basado en el algoritmo genético simple que posee varias características. (i) Almacena las soluciones no dominadas encontradas en una población externa, las cuales junto con los individuos de la población genética participan del proceso de selección. (ii) Asigna un

valor de adaptabilidad escalar a los individuos basado en el concepto de dominancia Pareto, el cual también ayuda a preservar la diversidad en la población genética. (iii) Mantiene constan te el número soluciones no dominadas almacenadas en la población externa sin destruir las características del frente aproximado.

En cada generación del método SPEA, los individuos de la población genética no dominados por los elementos de la población externa son copiados en esta última, eliminándose las soluciones que resulten dominadas. De esta forma, se preservan las buenas soluciones encontradas durante el proceso de búsqueda, a lo cual se denomina *elitismo*.

Debido a que la población externa interviene en los procesos de selección y de asignación del valor de adaptabilidad a los individuos de la población genética, esta debe limitarse a cierto tamaño de manera a que no disminuya la presión de selección provocando que la búsqueda de soluciones se ralentice [Zitzler98]. Así, cuando el tamaño de la población externa supera

cierto limite, esta se reduce aplicando un procedimiento denominado *clustering*, que conserva las características del frente Pareto aproximado hasta el momento [Zitzler98] y cuyo pseudocódigo se ilustra en la figura 2.14.

**Fig. 3** Pseudocódigo del procedimiento de clustering.

1. Recibir población externa de elementos no dominados E;

2. max\_elem := Recibir número máximo de elementos para E;

3. /\* se inicializa un conjunto de clusters \*/

4. cluster\_set := Ø;

5. FOR cada elemento P en E DO

6. cluster\_set := cluster\_set U {{P}};

7. END\_FOR

8. WHILE |cluster\_set| > max\_elem DO

9. min\_dist = 8;

10. FOR cada par (X, Y) en cluster\_set DO

11. IF distancia\_entre(X, Y) < min\_dist THEN

12. cluster\_1 := X;

13. cluster\_2 := Y;

14. min\_dist := distancia\_entre(X, Y);

15. END\_IF

16. END\_FOR

17. nuevo\_cluster := cluster\_1 U cluster\_2;

18. cluster\_set := cluster\_set / {cluster\_1, cluster\_2};

19. cluster\_set := cluster\_set U nuevo\_cluster;

20. END\_WHILE

21. E := Ø;

22. FOR cada elemento c en cluster\_set DO

23. P := Obtener individuo central en C;

24. E := E U P;

25. END\_FOR

26. Retornar nuevo conjunto de elementos no dominados E;

El valor de adaptabilidad para cada individuo de la población externa se halla dividiendo el número de individuos a los que domina en la población genética por el tamaño de la población genética incrementado en uno. El valor de adaptabilidad de cada individuo de la población externa se obtiene calculando la inversa de su valor de *strength*.

El valor de adaptabilidad de cada individuo de la población genética también se obtiene a través de la determinación de su valor de *strength*, el cual se halla sumando los valores de *strength* de los individuos de la población externa que lo dominan y luego incrementando el resultado de la suma en uno. Nuevamente, el valor de adaptabilidad de cada individuo es igual a la inversa de su valor de *strength*.

Mediante el procedimiento de cálculo del valor de adaptabilidad en la población externa, los individuos en dicha población que dominan a muchos individuos de la población genética reciben un bajo valor de adaptabilidad, con lo que se intenta evitar que a partir estos se generen nuevas soluciones que podrían resultar en la misma zona del espacio objetivo, no mejorándose así el frente Pareto aproximado hasta el momento [Zitzler98].

De la misma forma, los individuos de la población genética dominados por varios individuos de la población externa reciben un bajo valor de adaptabilidad, con lo que se intenta evitar que estos individuos den origen a nuevas soluciones que posiblemente resulten dominadas por las soluciones ya existentes [Zitzler98].

Luego de hallarse el valor de adaptabilidad para los individuos de la población externa y la población genética, se procede a identificar un conjunto de apareamiento (*mating pool*) a partir de la unión de estas poblaciones mediante una selección por torneo binario entre los individuos a través de sus valores de adaptabilidad, sobre los cuales posteriormente se aplican los operadores genéticos de cruzamiento y mutación.

Los individuos resultantes de la aplicación de las operaciones de cruzamiento y mutación sobre el *mating pool* constituyen la población para la siguiente generación.

El pseudocódigo genérico del método SPEA se muestra en la figura 4. Posteriormente este método fue mejorado por sus creadores [Zitzler02] al cual se denominó *Strength Pareto Evolutionary Algorithm 2* (SPEA2).

**Fig. 4** Pseudocódigo genérico del método SPEA.

1. t := 0;

2. Generar la población inicial P(0);

3. E := Ø; /\*Población externa\*/

4. DO

5. Evaluar P(t);

6. Actualizar población externa E;

7. Reducir el tamaño E mediante procedimiento de clustering;

8. M(t) := Seleccionar(P(t) U E); /\*mating pool\*/

9. P(t+1) := Cruzar y mutar M(t);

10. t := t+1;

11. WHILE(condición\_de\_parada() ==FALSE)

12. retornar soluciones no dominadas de E U P(t)

* + 1. Nondominated Sorting Genetic Algorithm II

El método NSGA [Srinivas94] posee varias falencias [Deb02]. (i) El procedimiento de clasificación de los individuos en varios frentes posee una alta complejidad computacional. (ii) El método no presenta elitismo lo cual degrada su rendimiento al perder las buenas soluciones *s* encontradas. (iii) Es necesaria la especificación a priori del valor para el parámetro *.share*

Con la intención de solucionar estos problemas Deb et al. [Deb02] presentan una versión mejorada de este método denominada *Nondominated Sorting Genetic Algorithm II* (NSGA2).

El primer problema es enmendado en NSGA2 con la utilización de un nuevo procedimiento de clasificación de los individuos de la población genética en varios frentes, denominado *fast nondominated sorting procedure*, el cual se desarrolla como sigue. Primeramente se determina para cada individuo *i* de la población el número de individuos que lo dominan *ND*[*i*] y el conjunto *D*[*i*] de todos los individuos a los que domina. Todos los individuos *i* cuyo valor de *ND*[*i*] sea igual a cero conforman en primer frente en la población.

Para todos los elementos en los conjuntos *D*[*i*] de los individuos del primer frente se decrementa en uno su valor de *ND*[*i*] y aquellos individuos para los cuales su valor de *ND*[*i*] resulte igual a cero constituyen el segundo frente en la población. Ahora se sigue el mismo proceso con los elementos en los conjuntos *D*[*i*] de los individuos del segundo frente para determinar el tercer frente en la población. Este proceso se repite hasta clasificar todos los elementos de la población en sus respectivos frentes. El pseudocódigo para el *fast nondominated sorting procedure* se muestra en la figura 5.

1. Recibir población de individuos P;

2. FOR cada individuo p en P DO

3. D[p] := Ø;

4. FOR cada individuo q en P DO

5. IF P DOMINA q THEN

6. D[p] := D[p] U q;

7. ELSE IF q DOMINA p THEN

8. ND[p] := ND[p] + 1;

9. END\_IF

10. END\_FOR

11. IF ND[p] := 0 THEN

12. Frente[1] := Frente[1] U p;

13. END\_IF

14. END\_FOR

15. f := 1

16. WHILE (Frente[f] != Ø)

17. H := Ø;

18. FOR cada p en Frente(f) DO

19. FOR cada q en D[p]DO

20. ND[q] := ND[q] - 1;

21. IF(ND[q] = 0) THEN

22. H := H U q;

23. END\_IF

24. END\_FOR

25. f := f + 1;

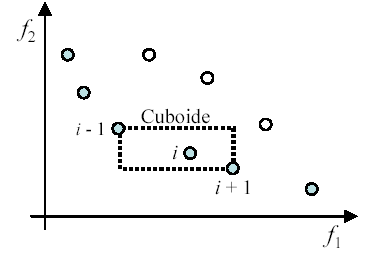
26. Frente[f] := H;

27. END\_WHILE

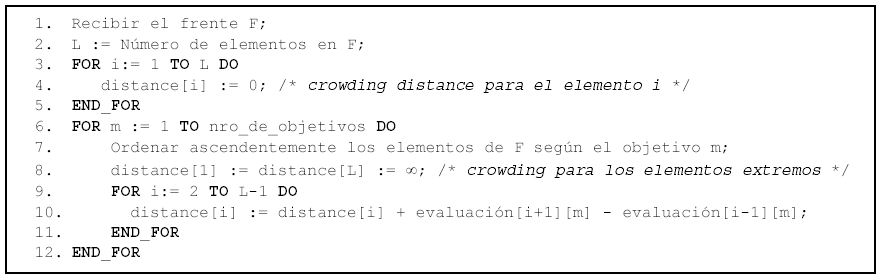
28. Retornar los Frentes 1 a f-1;

**Fig. 5** Pseudocódigo para el fast nondominated sorting procedure.

El tercer problema de NSGA se soluciona en NSGA2 a través de un procedimiento que permite obtener una estimación de la densidad de soluciones alrededor de una solución en el espacio objetivo sin la necesidad de especificar el valor de algún parámetro para ello. Este procedimiento calcula para cada elemento *i* en cada frente un valor que indica la longitud lateral promedio del cuboide más grande que encierra a la solución *i* sin incluir a otra solución de su frente, este valor se denomina *crowding distance* y se denota como *distance*[*i*]. La figura 6 ilustra este caso en un espacio solución bidimensional. El pseudocódigo para el cálculo del *crowding distance* de cada individuo en un frente dado se muestra en la figura 7.



**Fig. 6** Cuboide para un individuo i en un espacio bidimensional.



**Fig. 7** Pseudocódigo para el cálculo del crowding distance de cada individuo en un frente F.

Para evitar perder las buenas soluciones encontradas y mantener la población genética uniformemente distribuida a lo largo del frente Pareto aproximado, en NSGA2 se define un operador de comparación de individuos denominado *crowded comparison operator* (=), el n cual guía al proceso de selección de los individuos definiendo un orden parcial (de mejor a peor) entre estos y se define como:

*I* = *j* si (*frente*[*i*] < *frente*[*j*]) o ((*frente*[*i*] = *frente*[*j*]) y (*distance*[*i*] > *distance*[*j*])) , n donde *frente*[*i*] indica el número del frente en el que se encuentra el individuo *i* y *distance*[*i*] indica el *crowding distance* del individuo *i.*

El pseudocódigo genérico para el método NSGA2 es presentado en la figura 8. Primeramente, una población inicial *P*(0) de *n* individuos es creada y clasificada en los varios frentes mediante el *fast nondominated sorting procedure*. A cada individuo de la población se le asigna un valor de adaptabilidad igual a número de frente en que se encuentra y de esta forma se asume que lo deseado es la minimización del valor de adaptabilidad de los individuos.

A partir de este punto se inician las generaciones del algoritmo. Se realiza una selección por torneo binario de individuos de la población P(*t*) a través de su valor de adaptabilidad para la generar una población *P’*(*t*) de *n* individuos mediante la aplicación de las operaciones de recombinación y mutación.

**Fig. 8** Pseudocódigo genérico del método NSGA2.

1. t := 0;

2. Generar la población inicial P(0) de n individuos;

3. Clasificar P(0) mediante fast nondominated sorting procedure;

4. DO

5. P’(t) := seleccionar, cruzar y mutar P(t);

6. R(t) := P(t) U P’(t);

7. Clasificar R(t) mediante fast nondominated sorting procedure;

8. P(t+1) := Ø;

9. f := 1;

10. WHILE (P(t+1) < n) DO /\* n es el tamaño de la población \*/

11. Calcular crowding distance para el frente(f) de R(t);

12. Agregar individuos del frente(f) de R(t) a P(t+1);

13. END\_WHILE

14. Ordenar P(t+1) mediante el crowded comparison operator;

15. Reducir tamaño de P(t+1) a n elementos, eliminando los últimos elementos;

16. t := t+1;

17. WHILE(condición\_de\_parada() == FALSE);

18. Retornar soluciones no dominadas de P(t);

La poblaciones *P*(*t*) y *P’*(*t*) son combinadas en una única población *R*, la cual es clasificada en varios frentes mediante el *fast nondominated sorting procedure*. Luego se halla el valor de *crowding distance* para los individuos de cada frente de *R* y se copian los individuos de cada frente en la población para la siguiente generación *P*(*t*+1) hasta que se supere el número de *n* elementos. Nótese que los individuos de la población *P*(*t*+1) ya tienen especificados sus valores de adaptabilidad debido a que ya fueron clasificados en sus correspondientes frentes.

Finalmente, los elementos en la población *P*(*t*+1) son ordenados mediante el *crowded comparison operator* y se limita el tamaño de la población *P*(*t*+1) a *n* elementos, tomando los *n* primeros (mejores) elementos.

* + 1. Operadores Geneticos Utilizados por los Algoritmos MOEAs

Los algoritmos desarrolados utiliza los siguientes operadores genéticos:

* Operador de Selección: BinaryTournament
* Operador de Cruzamiento: TwoPointsCrossover
* Operador de Mutación: SwapMutation

1. Resultados Experimentales
   1. Descripción del Hardware Utilizada

Todos los algoritmos fueron implementados en Java (v. 1.6) y fueron ejecutados en un entorno Windows, Version Vista, en una máquina AMD Turion 2.2GHz con 3GB de memoria. Se realizaron diez corridas de 10 iteraciones para cada algoritmo y para cada problema de prueba. Como problemas de prueba se utilizaron dos instancias de cada tipo de problema (TSP, QAP y VRPTW). En el caso del TSP se utilizaron las instancias bi-objetivas de 100 ciudades KROAB100 y KROAC100. Para el QAP bi-objetivo, se utilizaron las instancias de 75 localidades qapUni.75.0.1 y qapUni.75.p75.1. Para el VRPTW bi-objetivo se utilizaron las instancias de 100 clientes c101 y rc101.

Se utilizó *m*=10 hormigas, =1, =2, =0.1, =0.8, =0.9, =0.5 , =1.

* 1. Métricas de Comparación

Para poder evaluar los resultados obtenidos en cada corrida de los métodos MOEAs

y MOACOs fueron utilizadas las métricas propuestas por Zitzler et al.

[Ziztzler00], que evalúan respectivamente la calidad de las soluciones, la distribución de las soluciones y la extensión del frente Pareto aproximado *Y’* devuelto en cada corrida. También los métodos fueron comparados con respecto al número de soluciones no dominadas encontradas en cada corrida, denotado por *|Y’|*.

La métrica proporciona una idea de la aproximación al frente Pareto real de un frente Pareto aproximado *Y’*, calculando el promedio de las distancias euclidianas de cada solución en el frente *Y’* a la solución más cercana en el frente .

La métrica estima la distribución promedio de las soluciones a lo largo de un frente Pareto aproximado *Y’*, calculando el número promedio de soluciones que se encuentran separadas de cada solución a una distancia mayor que cierto valor definido a priori.

La métrica evalúa la extensión o abarcamiento de un frente Pareto aproximado *Y’* a través de la sumatoria de las máximas separaciones de las evaluaciones en cada objetivo.

La métrica de cantidad de soluciones *|Y’|* da una idea acerca de la diversidad de combinación de las evaluaciones de los objetivos presentadas al Tomador de Decisiones, esta métrica puede ser considerada como un complemento de las demás métricas.

En la fig. 1 se puede apreciar las métricas. La definición formal de dichas métricas es:

donde *Y’* es el frente Pareto aproximado devuelto en una corrida, *d(p, q)* calcula la distancia euclidiana entre las soluciones *p* y *q*, |·| representa la cardinalidad, *M* es la dimensión del espacio objetivo y se estableció al 10% de la distancia entre los puntos extremos del frente Pareto aproximado *Y’*.

Para cada corrida, los valores de sus evaluaciones en cada métrica fueron normalizados a un número en el intervalo [0, 1], de manera a poder utilizar estos resultados en rankings de métodos presentados en la siguiente sección.

Las evaluaciones en la métrica fueron normalizadas restando de 1 el resultado de la división de la evaluación de cada corrida por el mayor valor obtenido en esta métrica en cada problema.

Para la métrica , la evaluación máxima de una corrida es igual al número de soluciones no dominadas encontradas en dicha corrida [Zitzler00], así las evaluaciones de las corridas fueron normalizadas dividiéndolas por el número de soluciones encontradas en dichas corridas.

Con relación a la métrica , las evaluaciones de cada corrida fueron normalizadas dividiéndolas por el valor de evaluación de esta métrica para el frente Pareto real de cada problema.

La cantidad de soluciones no dominadas encontradas *|Y’|* en cada corrida fue normalizada dividiéndola por el mayor valor de evaluación de esta métrica en cada problema.

De esta forma, los valores de evaluación normalizados son siempre menores que 1 y se consideran mejores cuanto más próximos encuentren a dicho valor.



Fig. Ejemplo de criterios (a) calidad (b) distribución (c) extensión para la comparación de los frentes Paretos aproximados [Lima07]

* 1. Resultados de la Comparación

El frente Ytrue conocido de cada problema fue generado previamente tomando las soluciones no dominadas generadas por todos los algoritmos en todas las corridas. Las tablas mostradas más abajo, armadas tomando cada Y’ de cada problema resuelto con los algoritmos MOEA y MOACOS respectivamente, presentan los resultados de las evaluaciones de las métricas aplicadas a los diferentes Paretos generados. Cada valor Y’ fue generado en base a las 10(diez) corridas del problema en cada algoritmo, tomando solamente las soluciones no dominadas de las mismas, obteniendo así un valor Y-true “parcial” para dicho algoritmo. Hemos tomado esta forma de promediar las corridas de un algoritmo debido a que refleja los mejores resultados de cada corrida.

Además, se muestran los gráficos correspondientes a los frentes Pareto generados por los distintos algoritmos para cada problema de prueba. Para las graficas, en el problema VRPTW, no se tomaron en cuenta las métricas M2’ y M3’, que evalúan la distribución y extensión del frente obtenido, debido a que los frentes Pareto encontrados contaban con escasa soluciones no dominadas, razón por la cual no se justifica realizar dichas métricas; pues, según los resultados pareciera ser que todos los algoritmos tienen una pésima distribución (M2’) y una excelente extensión (M3’) según se puede apreciar en la tabla xxxx. El lector puede notar que esto es falso ya que es imposible realizar estas concluisiones debido a que se cuenta con apenas uno o dos soluciones a lo sumo.

* + 1. TSP

|  |  |  |  |
| --- | --- | --- | --- |
| Algoritmos |  |  |  |
| MOACS | 0.97 | 0.98 | 0.91 |
| M3AS | 0.99 | 0.98 | 0.97 |
| NSGA | 0.7 | 0.97 | 0.56 |
| SPEA | 0.0 | 0.97 | 0.47 |

Tabla Resultado de las métricas sobre el TSP (KROAB)

La Tabla xxx muestra los resultados de las tres metricas para el problema de Cajero Viajante (TSP) y la Figura xxx muestran la representación de los paretos formados para las instancia de TSP KROAB100. Se puede apreciar que el mejor Algoritmo es el M3AS para M1’ y M2’ y el mejor para M3’ es el M3AS. Se nota a simple vista que los Paretos obtenidos por los MOEAS son malos en cuanto a distancia y extensión, aunque en cuanto a districución estan muy cerca de los MOACOS. Además se puede notar que el Ytrue queda totalmente solapado por los Paretos generado por M3AS y MOACS, esto se explica por el hecho que los mejores resultados para el Ytrue fueron hallados por estos algoritmos.

Fig. Frentes Pareto de los distintos algoritmos para el KROAB.

|  |  |  |  |
| --- | --- | --- | --- |
| Algoritmos |  |  |  |
| MOACS | 0.97 | 0.98 | 0.92 |
| M3AS | 0.98 | 0.98 | 0.93 |
| NSGA | 0.0 | 0.96 | 0.36 |
| SPEA | 0.01 | 0.97 | 0.47 |

Tabla Resultado de las métricas sobre el TSP (KROAC)

Fig. Frentes Pareto de los distintos algoritmos para el KROAC.

* + 1. QAP

|  |  |  |  |
| --- | --- | --- | --- |
| Algoritmos |  |  |  |
| MOACS | 0.01 | 0.83 | 0.18 |
| M3AS | 0.0 | 0.9 | 0.34 |
| NSGA | 0.94 | 0.97 | 0.89 |
| SPEA | 0.97 | 0.97 | 0.87 |

Tabla Resultado de las métricas sobre el QAP (qapUni.75.0.1.qap)

Fig. Frentes Pareto de los distintos algoritmos para el qapUni.75.0.1.qap.

|  |  |  |  |
| --- | --- | --- | --- |
| Algoritmos |  |  |  |
| MOACS | 0.0 | 0.66 | 0.07 |
| M3AS | 0.09 | 0.0 | 0.0 |
| NSGA | 0.92 | 0.97 | 1.0 |
| SPEA | 0.96 | 0.96 | 1.0 |

Tabla Resultado de las métricas sobre el QAP (qapUni.75.p75.1.qap)

Fig. Frentes Pareto de los distintos algoritmos para el qapUni.75.p75.1.qap.

* + 1. VRPTW

|  |  |  |  |
| --- | --- | --- | --- |
| Algoritmos |  |  |  |
| MOACS | 0.28 | 0.0 | 1.0 |
| M3AS | 0.0 | 0.0 | 1.0 |
| NSGA | 1 | 0.0 | 1.0 |
| SPEA | 0.14 | 0.0 | 1.0 |

Tabla Resultado de las métricas sobre el VRPTW (c101)

Fig. Frentes Pareto de los distintos algoritmos para el VRPTW (c101).

|  |  |  |  |
| --- | --- | --- | --- |
| Algoritmos |  |  |  |
| MOACS | 0.0 | 0.66 | 0.07 |
| M3AS | 0.09 | 0.0 | 0.0 |
| NSGA | 0.92 | 0.97 | 1.0 |
| SPEA | 0.96 | 0.96 | 1.0 |

Tabla Resultado de las métricas sobre el VRPTW (rc101).

Fig. Frentes Pareto de los distintos algoritmos para el VRPTW (c101).

1. Conclusiones
2. Trabajos Futuros
3. Referencias

|  |  |
| --- | --- |
| [Paciello06] | J. Paciello, H. Martínez, C. Lezcano and B. Barán. Algoritmos de Optimización multi-objetivos basados en colonias de hormigas. Proceedings of CLEI’2006. Latin-American Conference on Informatics (CLEI). Santiago, Chile. |
| [Pinto05] | D. Pinto y B. Barán. “Solving Multiobjective Multicast Routing Problem with a new Ant Colony Optimización approach”. LANC’05, Cali, Colombia. 2005. |
| [Knowles013 | J. Knowles y D. Corne. “Instance generators and test suites for the multiobjective quadratic assignment problem”. In: Fonseca, C.M., et al. Editors. Proc of EMO '03, LNCS 2632 page 295-310, Springer-Verlag, 2003 |
| [Garcia04] | C. García-Martínez, O. Cordón y F. Herrera. “An Empirical Análisis of Multiple Objective Ant Colony Optimización Algorithms for the Bi-criteria TSP”. ANTS Workshop 61-72. 2004 |
| [Baran03] | B. Baran y M. Schaerer. “A multiobjective Ant Colony System for Vehicle Routing Problems with Time Window*s”.* Proc. Twenty first IASTED International Conference on Applied Informatics, pg. 97-102. Insbruck, Austria. 2003 |
| [Deb99] | K. Deb. “Evolutionary Algorithms for Multi-Criterion Optimización in Engineering Design”. In Proceedings of Evolutionary Algorithms in Engineering and Computer Science EUROGEN’99. 1999 |
| [Coello99] | C. Coello. An updated Survey of Evolutionary Multiobjective Optimización Techniques: state of the art and future trends. In Congress on Evolutionary Computation. Piscataway, N. J., IEEE Service Center. 3–13. 1999 |
| [Gambardella99] | L. Gambardella, E. Taillard y G. Agazzi. “MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing  Problems with Time Windows”. In D. Corne, M. Dorigo, F. Glover (Eds.), New Ideas in Optimización, McGraw-  Hill, 73-76. 1999 |
| [Ziztzler00] | E. Zitzler, K. Deb and L. Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation, vol. 8, no.2, pp 173–195. 2000 |
| [Lima07] | J. Lima. Optimización de enjambre de partículas aplicada al problema del cajero viajante bi–objetivo, p. 87. 2007 |