TSP, QAP, VRPTW:  
Resolución y comparación 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 acerca de las implementaciones y evaluaciones 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 como el M-MMAS[9], el MOACS [8], el SPEA [7] y el NSGA [7].

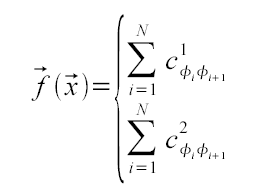
Se realizaron pruebas con dos instancias de cada problema. Se utilizaron reconocidos problemas de prueba de optimización multi-objetivo, el *Quadratic Assignment Problem* (QAP), definido en [Knowles03], el *Traveling Salesman Problem* (TSP) [5] y el *Vehicle Routing Problem with Time Windows* (VRPTW) [1]. 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 describen los algoritmos multi-objetivos utilizados. Los resultados experimentales de la comparación se muestran en la sección 4, y finalmente en la sección 5 se presentan algunas conclusiones y trabajos futuros en la sección 7.

1. Formulación de los problemas
   1. TSP

El TSP o problema del cajero viajante puede ser representado como un grafo completo G=(N, Ar), donde N es el numero de nodos, o ciudades, y Ar es el conjunto de arcos que interconectan completamente los nodos. Cada arco (i,j) є Ar posee un valor cij que representa la distancia entre ciudad i y la ciudad j. El problema consiste en encontrar el camino más corto que visite todas las ciudades exactamente una vez y vuelva a la ciudad de origen. Este camino se denomina camino Hamiltoniano, y por ende el TSP puede ser definido como encontrar el camino Hamiltoniano mas corto. Para los TSP simetricos, la distancia cij = cji. La definición detallada del problema puede encontrarse en [Gomez2004] y pertenece al conjunto de problemas NP-completos [Sahni1976].

En el caso *bi-objetivo*, se asocia a cada arco (i,j) un par de valores, uno para cada objetivo, que representa las distancias entre las ciudades i y j de acuerdo a cada objetivo. El problema consiste en minimizar:



Donde minimizar se refiere a la optimalidad Pareto y  representa la i-esima ciudad visitada. De manera a agregar el costo de retorno a la ciudad de origen, se considera a esta como la siguiente, luego de la N-esima ciudad visitada.

El bi-objetive TSP posee aplicaciones practicas en la optimización de rutas considerando, por ejemplo, tiempo de viaje y distancia recorrida.

* 1. QAP

El QAP o problema de asignación cuadrática es un problema NP-completo y consiste en asignar un conjunto de instalaciones a un conjunto de localidades con distancias conocidas entre cada par de localidades, dados los flujos entre cada para de instalaciones, de manera a minimizar la suma del producto entre los flujos y las distancias.

El QAP multi-objetivo (mQAP), propuesto por Knowles y Corne utiliza diferentes matrices de flujo, y mantiene la misma matriz de distancia. Dadas n instalaciones y n localidades, una matriz *A*={*aij* } є ℝ*n*×*n* donde *aij* representa la distancia entre las localidades *i* y *j*, y *b* matrices *Bq*={*bijq* } є ℝ*n*×*n , q*=1,... *, b* donde *bijq* representa el *q*-ésimo flujo entre las instalaciones ubicadas en las localidades *i* y *j,* el mQAP se define como:



donde minimizar se refiere a la optimalidad Pareto. La aplicabilidad real de este problema aparece en la ubicación de hospitales e instituciones sociales. Para este trabajo se consideró el bQAP (QAP bi-objetivo).

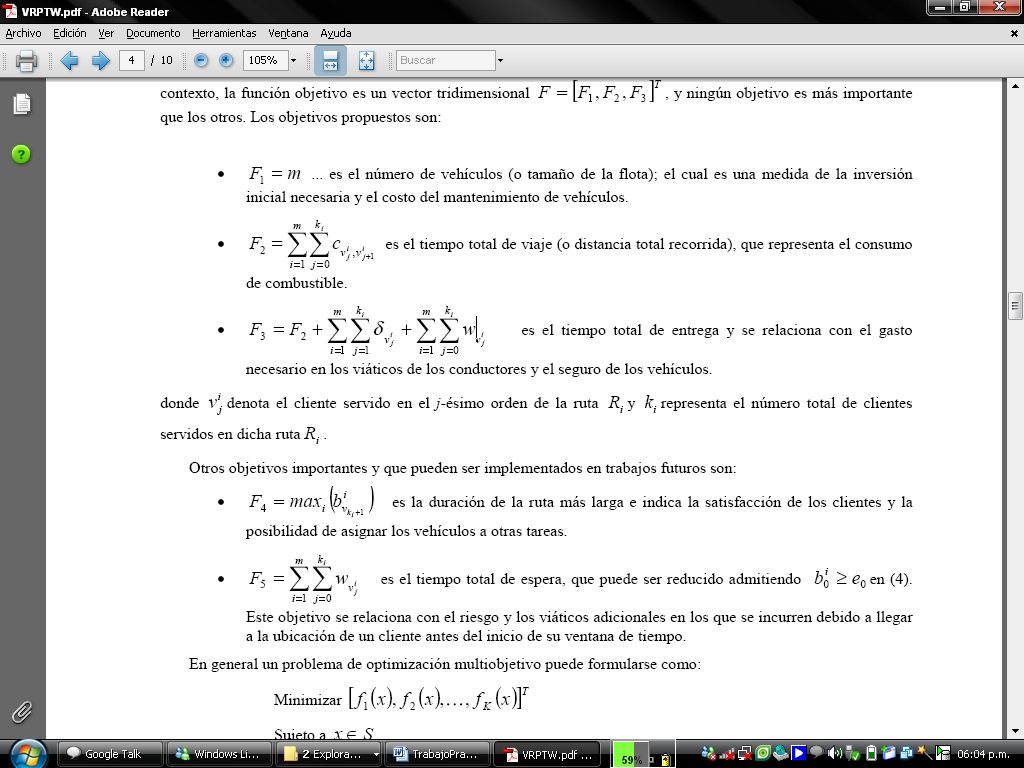
* 1. VRPTW

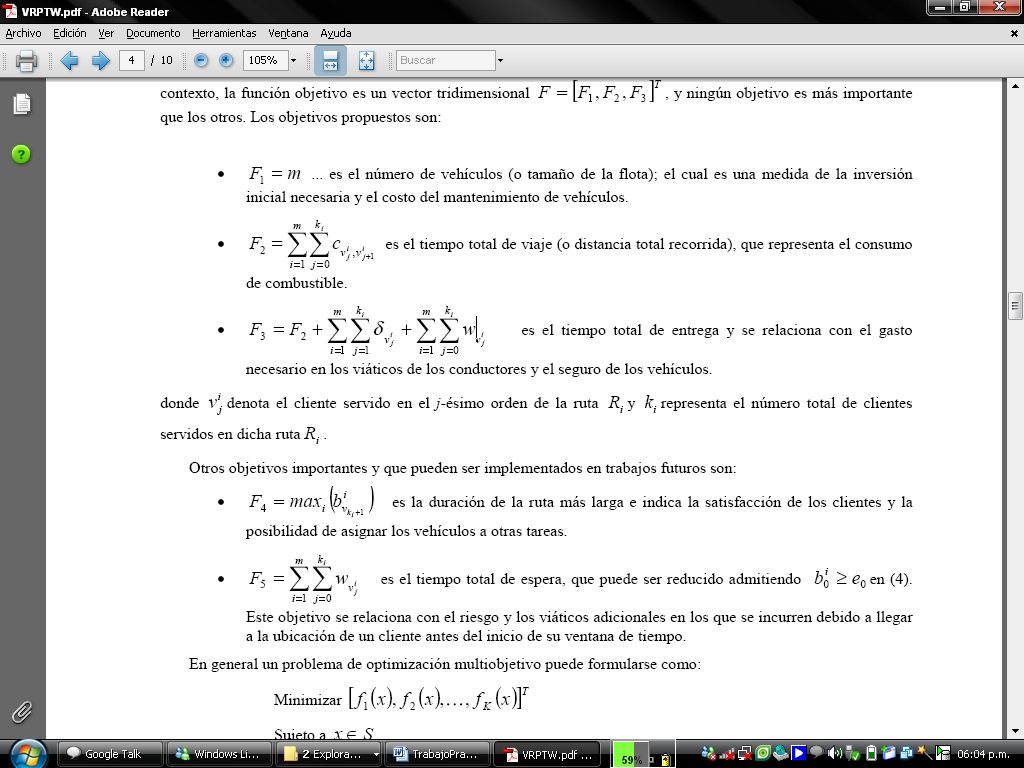
El objetivo del VRP es entregar bienes a este conjunto de clientes con demandas conocidas, al mínimo coste, encontrando las rutas óptimas que se originan y terminan en el referido depósito. Cada cliente es servido una sola vez y todos los clientes deben ser atendidos, para lo cual se los asigna a vehículos que llevarán la carga (demanda de los clientes que visitará) sin exceder su capacidad máxima de transporte.

Para extender el VRP tradicional agregando la restricción adicional de asociar una ventana de tiempo a cada cliente, se define un intervalo en el que cada cliente debe ser atendido y se obtiene el problema del ruteo de vehículos con ventanas de tiempo (Vehicle Routing Problem with Time Windows, o VRPTW) [5].

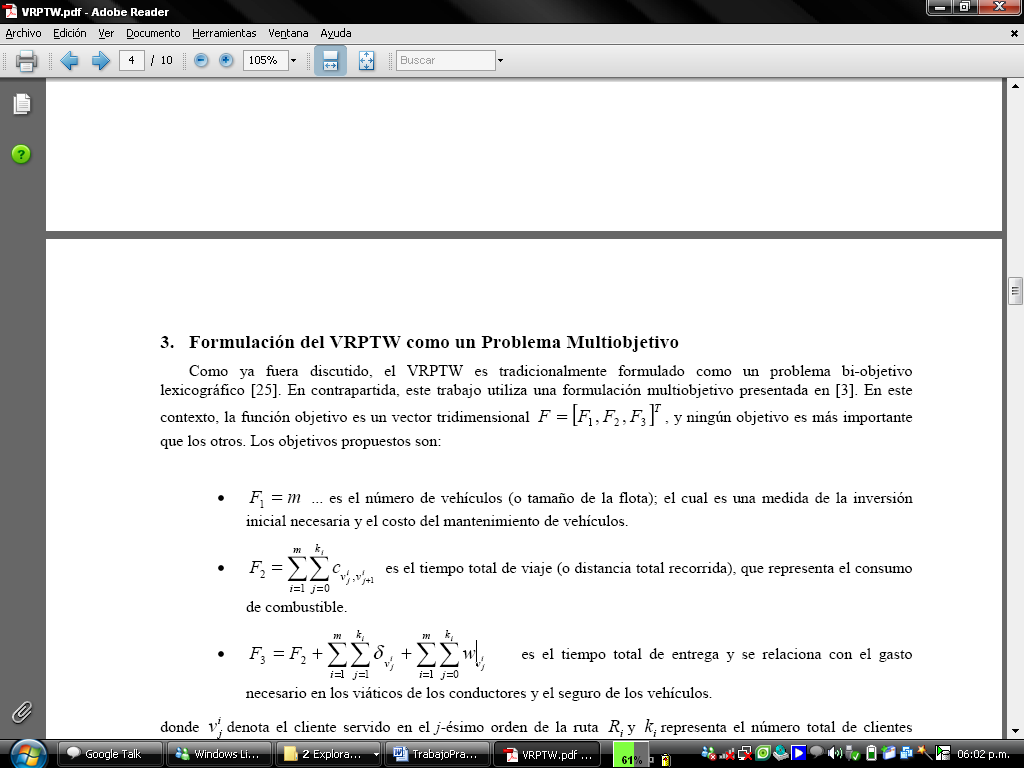
* + 1. Formulación del VRPTW como un Problema Multiobjetivo

Como ya fuera discutido, el VRPTW es tradicionalmente formulado como un problema bi-objetivo lexicográfico. En contrapartida, este trabajo utiliza una formulación multiobjetivo. En este contexto, la función objetivo es un vector tridimensional *F =* *F1, F2, F3**T* y ningún objetivo es más importante que los otros. Los objetivos propuestos a optimizar son:

es el número de vehículos (o tamaño de la flota); el cual es una medida de la inversión inicial necesaria y el costo del mantenimiento de vehículos.



es el tiempo total de viaje (o distancia total recorrida), que representa el consumo de combustible.

es el tiempo total de entrega y se relaciona con el gasto necesario en los viáticos de los conductores y el seguro de los vehículos.

Donde  ** denota el cliente servido en el *j*-ésimo orden de la ruta *R i* y *k i* representa el número total de clientes servidos en dicha ruta  *R i.*

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 [2] y [3].

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
   1. Algoritmos basado en colonia de hormigas

En la Fig. 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=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 [1], es una extensión del MACS-VRPTW, este último propuesto por Gambardella et al. [4]. 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 [9], 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 [9] 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. 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.

A continuación, en la Fig. 2 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).

* + 1. Strength Pareto Evolutionary Algorithm

El SPEA utiliza un archivo que contiene las soluciones no dominadas encontradas (población externa de no dominados Pnd). En cada generación, se copian los individuos no dominados de P a Pnd y se borra de este las soluciones dominadas. Para cada individuo en el sistema externo, se computa un valor de fuerza (strength) es proporcional al número de las soluciones a las cuales cada individuo domina.

En SPEA, el fitness de cada miembro de la población actual se computa según las fuerzas de todas las soluciones no dominadas externas que la dominen.

SPEA2 tiene las siguientes diferencias principales con respecto a su precursor: (1) incorpora una estrategia fina de asignación del fitness que considera, para cada individuo, el número de los individuos que lo dominan y el número de los individuos por los cuales es dominado; (2) utiliza la técnica del “vecino más cercano” para la valoración de la densidad, dirigiendo la búsqueda en forma más eficiente.

* + 1. Nondominated Sorting Genetic Algorithm

Se basa en la clasificación de individuos en varias capas o frentes. La clasificación consiste en agrupar a todos los individuos no dominados en un frente, con un valor de fitness (o adaptabilidad) igual para todos los individuos. Este valor es proporcional al tamaño de la población, para así proporcionar un potencial reproductivo igual para todos los individuos de este frente. Entonces el grupo de individuos clasificados es ignorado y otro frente de individuos no dominados es considerado. El proceso continúa hasta que se clasifican a todos los individuos en la población. Puesto que los individuos en el primer frente tienen el valor de fitness mayor, consiguen siempre más copias que el resto de la población.

DEB et al. propusieron una versión revisada del NSGA, llamada NSGA2, que es computacionalmente más eficiente. Además, es elitista y no necesita especificar ningún parámetro adicional. El NSGA2 no utiliza una memoria externa como los algoritmos anteriores (SPEA y SPEA2). El mecanismo elitista consiste en elegir los mejores. P. individuos de la unión de las poblaciones padre e hijo.

* + 1. Operadores Genéticos Utilizados por los Algoritmos MOEAs

Los algoritmos desarrolados utilizan los siguientes operadores genéticos:

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

1. Resultados Experimentales
   1. Hardware Utilizado

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 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.

Para los algoritmos MOACOs se utilizaron: *m*=10 hormigas, =1, =2, =0.1, =0.8, =0.9, =0.5, =1.

Para los algoritmos MOEAs se utilizaron: Tamaño de población = 100; Cantidad de evaluaciones = 25000; Probabilidad de Mutación = 1/Cant. de variables.

* 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. [10], 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 visualizar los resultados en términos porcentuales.

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.

* 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. Se ha tomado esta forma de promedio 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 escasas 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 las tablas 5 y 6. El lector puede notar que esto es falso ya que es imposible realizar estas conclusiones 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)

Fig. 3 Frentes Pareto de los distintos algoritmos para el KROAB.

La Tabla 1 muestra los resultados de las tres métricas para el problema de Cajero Viajante (TSP) y la Figura 3 muestran la representación de los Paretos formados para las instancia KROAB100 de TSP. Se puede apreciar que el mejor Algoritmo es el M3AS para M1’ y M2’ y el mejor para M3’ es el M3AS.

|  |  |  |  |
| --- | --- | --- | --- |
| 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 2 Resultado de las métricas sobre el TSP (KROAC)

Fig. 4 Frentes Pareto de los distintos algoritmos para el KROAC.

La Tabla 2 y la Figura 4 muestran las mismas métricas para la instancia KROAC100 de TSP con los resultados semejantes al KROAB100. El M3AS sigue llevando la delantera en calidad (M1’) y NSGA es el peor en la misma. La distribución de los resultados esta casi parejo en todos los algoritmos, mientras que en la métrica de Extensión el algoritmo que mejor se comporta es nuevamente el M3AS.

En general, para el problema de TSP, se nota a simple vista que los Paretos obtenidos por los MOEAS son malos en cuanto a calidad y extensión, aunque en la métrica de Distribución están muy cercas 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 los resultados de estos algoritmos.

* + 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 3 Resultado de las métricas sobre el QAP (qapUni.75.0.1.qap)

Fig. 5 Frentes Pareto de los distintos algoritmos para el qapUni.75.0.1.qap.

La Tabla 3 muestra las métricas de los algoritmos para la instancia **qapUni.75.0.1.qap** del Problema de Asignación Cuadrática (QAP) y la Figura 5 muestra su correspondiente representación grafica. Al contrario de las métricas de TSP, en este problema los MOEAS tiene mejores resultados que los MOACOS. Se puede apreciar que en cuanto a *Calidad* el mejor algoritmo es el SPEA, seguido del NSGA, en cuanto a *Distribución* están muy próximos los MOEAS y los MOACOS, llevando la delantera los primeros. Y para la medida *Extensión* los MOEAS son mejores.

|  |  |  |  |
| --- | --- | --- | --- |
| 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 4 Resultado de las métricas sobre el QAP (qapUni.75.p75.1.qap)

Fig. 6 Frentes Pareto de los distintos algoritmos para el qapUni.75.p75.1.qap.

La Tabla 4 y la Figura 6 muestran los resultados de las métricas para la instancia **qapUni.75.p75.1.qap** delProblema de Asignación Cuadrática (QAP). Nuevamente los MOEAS llevan la delantera en las métricas de esta instancia debido a la escasez de soluciones para los MOACOS. Similar al TSP, donde el *Ytrue* está formado prácticamente por los mejores resultados de los MOACOS, en el problema de QAP el *Ytrue* está formado por los mejores resultado de los MOEAS.

* + 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 5 Resultado de las métricas sobre el VRPTW (c101)

Fig. 7 Frentes Pareto de los distintos algoritmos para el VRPTW (c101).

|  |  |  |  |
| --- | --- | --- | --- |
| Algoritmos |  |  |  |
| MOACS | 0.2 | 0.0 | 1.0 |
| M3AS | 0.0 | 0.0 | 1.0 |
| NSGA | 1.0 | 0.0 | 1.0 |
| SPEA | 0.1 | 0.0 | 1.0 |

Tabla 6 Resultado de las métricas sobre el VRPTW (rc101).

La Tabla 5 y Figura 7 muestran los resultados de las métricas para la instancia c101 y la Tabla 6 muestra los resultados de las métricas para la instancia rc101 del Problema de Rutas de Vehículos con Ventana de Tiempo (VRPTW). Debido a la escasez de soluciones optimas, no se forman los paretos (Figura 7) ya que hay una única solución que además es el *Ytrue*. Para la instancia c101 el mejor algoritmo es el NSGA para la métrica de calidad. Como se mencionó líneas arriba, no tiene sentido comparar las métricas M2’ y M3’. Para la instancia rc101 el mejor algoritmo para la métrica de distancia (M1’) es el SPEA. En general, para VRPTW los algoritmos genéticos tienen mejor comportamiento para la métrica de *calidad* que los algoritmos basados en colonia de hormigas.

1. Conclusiones

De forma empírica se verificó que los algoritmos genéticos tienen un comportamiento no deseable para la resolución de problemas TSP, mientras que serian los favoritos para resolver los problemas de QAP y VRPTW teniendo en cuenta solamente su proximidad al Pareto Real (Y*true*). Los MOACOS tienen excelentes medidas en extensión, calidad y distribución para la resolución del Problema del Cajero Viajante. No hemos podido constatar la razón de este comportamiento.

Se puede apreciar que debido a la complejidad de los problemas con soluciones multiobjetivos y su dificultad NP-difícil no es fácil encontrar un algoritmo genérico para resolverlos todos en forma óptima. Algunos algoritmos tienen mejor desempeño en las métricas que otros, dependiendo del tipo del problema. Se demuestra, además, que debido a la cantidad de restricciones del Problema VRPTW, y los resultados obtenidos, este es un problema NP-Dificil sin lugar a dudas.

No existe un único algoritmo óptimo para resolver los tres problemas, dependiendo de que se desee resolver, se escogería uno u otro algoritmo. A pesar de eso, estos enfoques algorítmicos se aproximan bastante bien a los resultados deseados en un tiempo razonable.

1. Trabajos Futuros

Según el análisis que se realizó en el trabajo propuesto surgieron los trabajos futuros a realizar sobre los mismos:

* Llegar a comparar los algoritmos MOACOs y MOEAs con más estancias de las propuestas en cada tipo de problema (entiéndase por tipo de problema TSP, QAP, y VRPTW), con el fin de poder llegar a dar un criterio de comparación general de los mismos.
* Realizar pruebas alterando los parámetros específicos de cada algoritmo, a los efectos de mejorar el comportamiento en la resolución de los problemas.
* Llegar a utilizar más métricas para realizar la comparación entre los algoritmos, entre ellos se podría mencionar HiperVolumen, Epsilon, GeneralizedSpread, GeneralizationalDistance, InvertedGenerationalDistance, Spread (Métricas que se encontraron en el Framework de JMetal), entre otras.

1. Referencias

|  |  |
| --- | --- |
| [1] | 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 |
| [2] | 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. |
| [3] | K. Deb. “Evolutionary Algorithms for Multi-Criterion Optimización in Engineering Design”. In Proceedings of Evolutionary Algorithms in Engineering and Computer Science EUROGEN’99. 1999 |
| [4] | 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 |
| [5] | 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 |
| [6] | 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, |
| [7] | J. Lima. Optimización de enjambre de partículas aplicada al problema del cajero viajante bi–objetivo, p. 87. 2007 |
| [8] | 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. |
| [9] | D. Pinto y B. Barán. “Solving Multiobjective Multicast Routing Problem with a new Ant Colony Optimización approach”. LANC’05, Cali, Colombia. 2005. |
| [10] | E. Zitzler, K. Deb and L. Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation, vol. 8, no.2, pp 173–195. 2000 |