Una nueva propuesta de Templado Simulado Multiobjetivo

Hugo Meyer[1] y Benjamín Barán[1,2]

[1] Facultad Politécnica, Universidad Nacional de Asunción
http://www.pol.una.py/
[2] Universidad Católica, Ntra. Sra. de la Asunción"
meyer.hugo@gmail.com, bbaran@pol.una.py

Abstract. This paper compares several multiobjective simulated annealing algorithms – MOSA, using diverse ZDT functions as a test bed. In addition, a new version of MOSA is proposed. This new version seeks to guide the search process towards the objectives that are further away from their optimal values. For this purpose, when generating a solution that is not comparable with the current solution, it is accepted with high probability if the new solution improves the objective which is farther from optimum. In addition, alternative ways of carrying out the perturbation of solutions to obtain new ones are also presented. The proposed MOSA presents clear benefits compared to classical simulated annealing algorithm, especially considering the distance to the Optimal Pareto Front.

Palabras Claves: *Multi-Objective Simulated Annealing* - MOSA, Optimización Multiobjetivo, Templado Simulado, Soluciones no-comparables, Distancia al Frente Pareto Óptimo.

1 Introducción

Los problemas de optimización Multiobjetivo intentan encontrar un vector (variable independiente) que simultáneamente optimice (maximice o minimice) múltiples variables dependientes (u objetivos). Los objetivos generalmente están en competencia, es decir, mejorar un objetivo empeora otro, por lo que no siempre resulta posible obtener una única solución al problema que sea totalmente mejor que las demás soluciones, como ocurre en la optimización Mono-Objetivo. Consecuentemente, en la optimización de múltiples objetivos generalmente se termina calculando un conjunto de soluciones óptimas de compromiso (*trade-off*) conocido como Conjunto Pareto [2]. Formalmente, un problema de optimización de múltiples objetivos puede definirse como:

$$\begin{aligned} &\text{Optimizar} \quad \vec{y} = \vec{f}(\vec{x}) = [f_1(\vec{x}) \quad f_2(\vec{x}) \quad \cdots \quad f_k(\vec{x})] \\ &\text{sujeto a} \quad \vec{e}(\vec{x}) = [e_1(\vec{x}) \quad e_2(\vec{x}) \quad \cdots \quad e_m(\vec{x})] \geq \mathbf{0} \\ &\text{donde} \quad \vec{x} = [x_1 \quad x_2 \quad \cdots \quad x_n] \in X \\ &\qquad \vec{y} = [y_1 \quad y_2 \quad \cdots \quad y_k] \in Y \end{aligned} \tag{1}$$

X es el dominio de definición del problema.

Y es el espacio objetivo.

A lo largo de este trabajo cuando hablamos de problemas de optimización, nos referimos exclusivamente a problemas de minimización.

Para hablar de la relación entre posibles soluciones de un problema de múltiples objetivos ($k > 1$), se debe introducir formalmente el concepto de *Dominancia Pareto*:

Dado un vector $\vec{y} = [y_1 \quad y_2 \quad \cdots \quad y_k] \in Y$, se dice que este domina a $\vec{v} = [v_1 \quad v_2 \quad \cdots \quad v_k] \in Y$ si y sólo si: (2)

$$\forall\, i \in \{1,..., k\},\ y_i \leq v_i \ y \ \exists\ j \in \{1,..., k\} \mid y_j < v_j$$

La base de prueba utilizada son las funciones ZDT y se comparan 4 MOSAs, donde el propuesto trata de orientar las soluciones no comparables hacía la solución menos favorecida hasta ese momento del proceso de recocido simulado para obtener mejores resultados.

Para referirnos al estado de dominancia entre dos soluciones A y B utilizamos la siguiente notación:

$$\begin{aligned} &A \succ B \rightarrow A \text{ domina o es mejor que } B \\ &A \prec B \rightarrow A \text{ es dominado por } B \\ &A \sim B \rightarrow A \text{ no es comparable con } B \end{aligned} \tag{3}$$

2 Templado Simulado

2.1 Consideraciones Iniciales

El Templado Simulado tradicional (en adelante *Simulated Annealing* o SA) es una heurística mono-objetiva generalmente utilizada para la optimización de problemas con diversos mínimos locales [11]. A partir de una solución inicial elegida generalmente al azar, un SA va generando aleatoriamente una nueva solución cercana a la solución actual (o en el entorno de esta). Dicha solución candidata es aceptada como buena si reduce el valor de la función de costo (considerando una minimización) o es aceptada conforme a una determinada probabilidad en caso contrario. La probabilidad de aceptación se irá reduciendo con el número de iteraciones y de acuerdo al grado de empeoramiento del costo [16], conforme a la siguiente ecuación:

$$p(\Delta f) = \left(\omega \frac{-\Delta f}{T} \right) \quad (4)$$

Δf = variación de energía entre el valor de la función objetivo actual y la solución candidata.
T = temperatura actual;
ω = factor de corrección.

En un contexto mono-objetivo, al valor $f(\vec{x})$ de una solución $\vec{x} \in X$ se la denomina *Energía de la Solución,* lo que en general se desea minimizar. La temperatura actual se representa mediante la letra T y la misma va decreciendo mientras el algoritmo avanza, hasta que se llega a un punto de congelamiento, en el cual la solución ya no mejora [8].

Por su parte, los métodos de *Hill Climbing* son bastante similares pues también escogen un punto inicial del espacio de búsqueda como solución actual, para luego perturbarlo, pero a diferencia de un SA, solo se acepta a la solución candidata si mejora la solución actual [13]. El problema de los algoritmos de *Hill Climbing* es que la solución final suele quedarse en un mínimo local, sin llegar a la solución óptima [13]. Sin embargo, un *Simulated Annealing* puede aceptar soluciones que empeoran la solución actual con una probabilidad dada en (**4**), lo que reditúa en una mejor exploración [16].

2.2 Algoritmo Multiobjetivo

La mayoría de los problemas del mundo real tienen múltiples objetivos que satisfacer a la vez, y por lo tanto, es deseable extender las técnicas del *Simulated Annealing* para la optimización Multiobjetivo. Una forma interesante de encarar la versión *Multi-Objective Simulated Annealing* (MOSA) es mediante la combinación de objetivos en una suma ponderada mediante pesos, como muestra la ecuación (**5**). De esta forma, se puede resolver un problema multiobjetivo de manera similar a un problema Mono-Objetivo, como se hace en [3], [9], [12], [15], [17], [18], [19].

$$E(\vec{x}) = \sum_{i=1}^{k} w_i \, f_i(\vec{x}) \qquad \text{donde generalmente: } \sum_{i=1}^{k} w_i = 1 \quad (5)$$

El problema que surge con la ecuación (**5**) y sus variantes [6], [12], [19] es la forma de elección de los pesos que deben aplicarse a cada objetivo. Con pesos predefinidos y estáticos, partes del *Frente Pareto* quedarían inaccesibles [4], así como ocurre con los algoritmos evolutivos basados en composición de objetivos. Por este motivo, muchos autores han propuesto una variedad de esquemas para adaptar los pesos durante el proceso de recocido simulado de forma a mejorar la exploración del frente de soluciones no-dominadas [10]. La reinicialización del algoritmo a partir de soluciones no dominadas elegidas randómicamente es una buena opción para colaborar con la exploración [17].

De las variantes que utilizan una función compuesta, la que se mostró más prometedora fue la propuesta [12] que considera la dominancia. Si la nueva solución candidata ($\vec{x}$') domina a la actual ($\vec{x}$), la primera es asignada como actual, y si la

solución actual domina a la candidata, se buscará de todas formas realizar la asignación de la nueva solución como actual conforme a la probabilidad dada en (4), proponiéndose la utilización del promedio de la diferencia en los valores objetivos como valor de Δf . Los algoritmos que utilizan funciones compuestas de energía se ven muy limitados en su cobertura del frente Pareto [4]. Por otro lado, es difícil obtener una prueba de convergencia de algoritmos que modifican su heurística a lo largo del tiempo para explorar de manera transversal el frente de soluciones no-dominadas. Estas debilidades hacen que no se desee utilizar algoritmos multiobjetivo con una función compuesta por varios objetivos.

Los MOSA actuales en lugar de tomar los objetivos y combinarlos en una sola ecuación, los tratan por separado y en el momento de realizar comparaciones se considera el concepto de dominancia definido en (**2**).

2.2 Implementaciones realizadas

El MOSA clásico que se presenta en el *Pseudocódigo 1* es una versión bi-objetiva que actualiza el *Frente Pareto* conocido (P_{known}) de acuerdo a las soluciones que va encontrando, y acepta soluciones dominadas conforme a (**4**).

```
Input: f(x), X , T0, Tf, L, α.
Output: Pknown
x = solución inicial, T = T0, Pknown = {x}
while (T > Tf)
      for i = 1 : L
            x' = Perturbar_Solución(x)
            if  (x' ≻ x  or  x' ~ x )
                  x = x'
                  Pknown = Actualizar_Pareto(x, Pknown)
            else if (uniforme(0,1) < exp(-Δf/T))
                  x = x'
            end
      end
      T = αT
end
```

Pseudocódigo 1. MOSA Clásico

Puede verse en este MOSA clásico que si una solución nueva domina o es no comparable con la solución actual, se busca actualizar el *Frente Pareto* utilizando el *Pseudocódigo 2.* La función uniforme(0,1) obtiene un valor aleatorio entre 0 y 1 para luego compararlo con el resultado obtenido conforme a (**4**).

Al momento de perturbar una solución para obtener otra solución candidata, se presentan dos posibles variantes: modificar una sola de las *k* componentes de $\vec{x}$ o modificar más de una componente, como se muestra en los *pseudocódigos 3* y *4*.

```
Input: x⃗ , Pknown
Output: Pknown
for i = 1 : size(Pknown)
      if( x⃗ ≻ Pknown[i] )
            Remover (Pknown[i])
      else if( x⃗ ≺ Pknown[i] )
            return()
      end
end
Añadir x⃗ a Pknown
return Pknown
```

Pseudocódigo 2. Actualizar_Pareto.

```
Input: x⃗ , X , T0, T
Output: x⃗ ′
li = limiteInferior( X ) * (T/T0)
ls = limiteSuperior( X ) * (T/T0)
p = indice_aleatorio( x⃗ )// p ∈ {1, ..., n}
perturbación = uniforme(li,ls)
if (uniforme(0,1) > 0.5)
      x⃗ ′[p] = x⃗ [p] + perturbación
else
      x⃗ ′[p] = x⃗ [p] - perturbación
end
return x⃗ ′
```

Pseudocódigo 3. Variante de perturbación que modifica un solo valor del vector $\vec{x}$.

```
Input: x⃗ , X , T0, T
Output: x⃗ ′
li = limiteInferior( X ) * (T/T0)
ls = limiteSuperior( X ) * (T/T0)
perturbación = uniforme(li,ls)/n
for i = 1 : n
     if (uniforme(0,1) > 0.5)
            x⃗ ′[i] = x⃗ [i] + perturbación
     else
            x⃗ ′[i] = x⃗ [i] - perturbación
     end
end
return x⃗ ′
```

Pseudocódigo 4. Variante de perturbación todos los valores de $\vec{x}$.

2.3 Algoritmo propuesto

El MOSA clásico propone que se acepte una solución nueva cuando esta domina o es no comparable con la actual. El algoritmo propuesto en este trabajo trata de manera diferente a las soluciones candidatas que no son comparables con la actual. En esta nueva variante, en caso de que dos soluciones sean no comparables, se busca realizar un cambio a la solución candidata con alta probabilidad si la misma nos permite mejorar el objetivo que en ese momento se encuentra más cerca de su peor valor, privilegiando los objetivos que todavía requieren mayor refinamiento. Con esta mejora, se busca direccionar el algoritmo a encontrar mejores soluciones para aquel objetivo que se ve menos beneficiado, como puede apreciarse en el *Pseudocódigo 5* que propone utilizar reinicializaciones a valores aleatorios del Frente Pareto como una forma de mejorar la exploración.

Corresponde mencionar que la función peorE presente en el *Pseudocódigo 5* devuelve el índice del objetivo que se encuentra más cercano a su peor valor. Entonces, si se genera una solución candidata ($\vec{x}$ ') que resulta ser no comparable con la solución actual ($\vec{x}$), se busca cambiarla dando prioridad al objetivo menos favorecido en ese momento. En caso de que estas condiciones no se den, de todas formas se trata de hacer que la solución candidata sea establecida como la actual, atendiendo a una probabilidad establecida experimentalmente en 60%.

3 Pruebas Realizadas y Resultados Experimentales

Diferentes características pueden hacer que el MOSA no converja al *Conjunto Pareto Óptimo* e influyan en la diversidad en las soluciones. El mantenimiento de la diversidad en la población de soluciones es necesario para la obtención de un *Frente Pareto* bien distribuido.

Como se mencionó más arriba, ciertas características del Frente Pareto pueden dificultar al algoritmo que converja correctamente, como por ejemplo [5]:

- Convexidad o No-convexidad.
- Discretitud.
- No-uniformidad.

```
Input: f(x⃗), X , T0, Tf, L, α
Output: Pknown
x⃗ =solución_inicial, Pknown = { x⃗ }, T = T0, indicePeorE=0,
peorE1 = x⃗ .energia1, peorE2 = x⃗ .energia2
while (T > Tf)
      Reinicialización_a_Pareto()
      for i = 1 : L
            x⃗ ′ = Perturbar_Solución( x⃗ )
            If  ( x⃗ '≻ x⃗ )
                  x⃗ = x⃗ '
                  Pknown = Actualizar_Pareto( x⃗ , Pknown )
            else if( x⃗ '≺ x⃗ )
                  if( x⃗ '.energia1 > peorE1 )
                      peorE1 = x⃗ '.energia1
                  else if( x⃗ '.energia2 > peorE2 )
                      peorE2 = x⃗ '.energia2
                  end
                  if (uniforme(0,1) < exp(-Δf/T))
                      x⃗ = x⃗ '
                  end
            else if (uniforme(0,1) > 0.2)
                  indicePeorE = peorE( x⃗ , peorE1, peorE2 )
                  if( x⃗ .energia1 > x⃗ '.energia1 & indicePeorE=1)
                        x⃗ = x⃗ '
                  else if( x⃗ .energia2 > x⃗ '.energia2 &
                  indicePeorE=2)
                        x⃗ = x⃗ '
                  end
                  if (uniforme(0,1) > 0.4)
                        x⃗ = x⃗ '
                  end
            end
      end
      T = αT
end
```

Pseudocódigo. 5. MOSA propuesto.

Para una evaluación adecuada de las alternativas para un MOSA, se escogió un conjunto de funciones de prueba que contemplan distintas posibilidades. Estas funciones consideran la minimización de dos objetivos y son llamadas *Funciones ZDT* [20]. Dichas funciones fueron utilizadas para realizar comparaciones entre el

MOSA clásico, el MOSA clásico con reinicializaciones a valores del Frente Pareto, el MOSA clásico con perturbación distribuida entre todos los valores de $\vec{x}$ (*Pseudocódigo 4*) y el MOSA propuesto en este trabajo (*Pseudocódigo 5*).

Se realizaron pruebas con las funciones ZDT1, ZDT2, ZDT3 y ZDT4. La métrica principal considerada en la evaluación fue:

- La distancia del conjunto de soluciones no dominadas resultante al *Frente Pareto Óptimo* [20].

Los resultados experimentales relevantes son mostrados en la Tabla 2 y 3, donde se utiliza la siguiente nomenclatura:

- MOSA 0: Algoritmo Clásico (*Pseudocódigo 1*) con perturbación de un solo valor (*Pseudocódigo 3*).
- MOSA 1: Algoritmo Clásico (*Pseudocódigo 1*) con perturbación de un solo valor (*Pseudocódigo 3*) y con reinicializaciones a valores del *Frente Pareto.*
- MOSA 2: Algoritmo Clásico (*Pseudocódigo 1*) con perturbación distribuida entre todos los valores de $\vec{x}$ (*Pseudocódigo 4*).
- MOSA 3: Algoritmo Propuesto (*Pseudocódigo 5*) con perturbación de un solo valor (*Pseudocódigo 3*).

En la Tabla 2 se muestran los valores obtenidos atendiendo a la métrica llamada *Distancia al Frente Pareto Óptimo*, pues en esta es donde el MOSA 3 muestra marcadas diferencias frente a los demás. En la Tabla 3 se muestran los resultados obtenidos realizando la unión de los Frentes Paretos de las diez corridas, considerando como *Frente Pareto Óptimo* a la unión de los diez frentes óptimos.

Los parámetros utilizados pueden observarse en la Tabla 1.

Tabla 1. Parámetros utilizados en las corridas de los diferentes algoritmos.

Corrida	Temperatura Inicial (T_0)	Temperatura Final (T_f)	Iteraciones
1	50,2125734	6,13E-04	1000
2	60,1577012	8,20E-05	1500
3	120,640779	7,46E-05	2000
4	19,5230757	4,40E-05	770
5	11,3433823	0,00978428	600
6	13,9283272	0,00848548	400
7	11,4489245	7,77E-04	500
8	13,7250506	4,23E-04	450
9	10,4994901	0,00515572	268
10	18,7248316	0,00753646	558

Tabla 2. Resultados obtenidos con las pruebas ZDT teniendo en cuenta la métrica: *Distancia al Frente Pareto Óptimo*. Los mismos representan el promedio de valores obtenidos atendiendo a las corridas a los parámetros presentados en la Tabla 1.

Función	MOSA 0	MOSA 1	MOSA 2	MOSA 3
ZDT1	0,045166	0,018718	0,621148	0,018614
ZDT2	0,075697	0,047704	0	0,029592
ZDT3	0,025723	0,018172	0,516336	0,016491
ZDT4	0,703925	0,242556	0,909677	0,204571

Tabla 3. Resultados obtenidos con las pruebas ZDT teniendo en cuenta la métrica: *Distancia al Frente Pareto Óptimo*. Los mismos fueron obtenidos realizando la unión de los *Frentes Paretos* de las diez corridas para cada algoritmo y comparados contra el *Frente Pareto Óptimo* obtenido también de esas diez corridas.

Función	MOSA 0	MOSA 1	MOSA 2	MOSA 3
ZDT1	0,027271	0,008563	0,701987	0,006527
ZDT2	0,063496	0,020708	0	0,01187
ZDT3	0,005504	0,005403	0,557792	0,003642
ZDT4	0,781739	0,463772	1,463035	0,142622

Como puede observarse el MOSA 3 es el que presenta mejores resultados considerando la métrica mencionada más arriba. En segundo lugar puede verse al MOSA 1 y presentando resultados cercanos a este el MOSA 0. El MOSA 2 queda en último lugar presentando resultados muy poco prometedores.

4 Conclusiones

Los Algoritmos Multiobjetivo de Templado Simulado (MOSA) que operan con funciones compuestas utilizando pesos han sido ya prácticamente desplazados por las implementaciones que trabajan con dominancia, definida en (**2**).

Se destaca que el MOSA 0 y el MOSA 1 presentan buenos valores considerando métricas como: *Error, Cantidad de Soluciones y Distancia al Frente Pareto Óptimo. D*e hecho el MOSA 0 presenta los mejores resultados considerando el *error* para los ZDT 1, 2 y 3, pero en el ZDT4 el MOSA 3 presenta resultados muy prometedores, lo que sugiere que a medida que se aumenta la complejidad, el mismo responderá de mejor manera. Considerando la métrica *Cantidad de Soluciones*, el MOSA 1 se comportó bastante bien, de manera casi idéntica al MOSA 3.

Considerando la *Distancia al Frente Pareto Óptimo* puede verse que el MOSA 3 propuesto en este trabajo es el que presenta los mejores resultados experimentales, observándose que orientando mínimamente el proceso hacia aquellos objetivos que requieren mayor optimización se produce una mejoría notable en todos los casos, por lo que resulta evidente las ventajas de utilizar el algoritmo propuesto.

Referencias

1. Bandyopadhyay S., Saha S., Maulik U., y Deb K. A Simulated Annealing-Based Multiobjective Optimization Algorithm: AMOSA. Ieee Transactions on Evolutionary Computation, vol. 12, no. 3, junio 2008.
2. Coello C.A. A Short Tutorial on Evolutionary Multiobjective Optimization. First International Conference on Evolutionary Multi-Criterion Optimization, págs. 21–40. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.
3. Czyzak P. y Jaszkiewicz A. Pareto simulated annealing – a metaheuristic technique for multipleobjective combinatorial optimization. J. Multi-Criteria Decision Anal., 34–47, 1998.
4. Das I. y Dennis J. A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems. Structural Optimization, 63–69, 1997.
5. Deb K. Multi-Objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems. Technical Report CI-49/98, Dortmund: Department of Computer Science/LS11, University of Dortmund, Germany, 1998.
6. Engrand P. A multi-objective approach based on simulated annealing and its application to nuclear fuel management. In 5th International Conference on Nuclear Engineering, páginas 416–423, Nice, Francia, 1997.
7. Fleischer M. Simulated Annealing: Past, Present and Future. Procedings of the 1995 Winter Simulation Conference. 1995
8. Geman S., Geman D. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1984.
9. Hapke M., Jaszkiewicz A., y Slowinski R. Pareto simulated annealing for fuzzy multi-objective combinatorial optimization. Journal of Heuristics, 6(3):329–345, 2000.
10. Jaszkiewicz A. Comparison of local search-based metaheuristics on the multiple objective knapsack problem. Foundations of Computer and Decision Sciences, 26(1):99–120, 2001
11. Kirkpatrick S., Gelatt C.D. y Vecchi M.P, Optimization by Simulated Annealing. Science, 1983.
12. Nam D. K. y Park C. H. Multiobjective simulated annealing: a comparative study to evolutionary algorithms. International Journal of Fuzzy Systems, 2(2):87–97, 2000.
13. Russell S., Norvig P. Artificial Intelligence: A Modern Approach (2nd ed.), Upper Saddle River, NJ: Prentice Hall, pp. 111-114, 2003.
14. Saleh M.A. y Fox G., (1998) A comparison of Annealing Techniques for Academic Course Scheduling. Practice and Theory of Automated Timetabling II, Selected Papers from the 2nd International Conference, PATAT 1997.
15. Serafini P. Simulated annealing for multiobjective optimization problems. In Multiple criteria decision making. Expand and enrich the domains of thinking and application, páginas 283–292, 1994.
16. Smith K., A Study of Simulated Annealing Techniques for Multi-Objective Optimisation. University of Exeter, octubre 2006.
17. Suppapitnarm A., Seffen K.A., Parks G.T., y Clarkson P.J. A simulated annealing algorithm for multiobjective optimization. Engineering Optimization, 33:59–85, 2000.
18. Tuyttens D., Teghem J., y El-Sherbeny N.. A particular multiobjective vehicle routing problem solved by simulated annealing. En Gandibleux X., Sevaux M., S¨orensen K., y T'kindt V., editors, Metaheuristics for multiobjective optimisation, volume 535 of Lecture Notes in Economics and Mathematical Systems, páginas 133–152, 2003.
19. Ulungu E.L., Teghaem J. , Ph. Fortemps, y D. Tuyttens. MOSA method: a tool for solving multiobjective combinatorial decision problems. Journal of Multi-criteria Decision Analysis, páginas 221–236, 1999.
20. Zitzler E., Deb K., y L. Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation, 8(2):173–195, 2000.