ESTUDIO DE VARIACIÓN POBLACIONAL USANDO HARMONY SEARCH PARA LA RESOLUCIÓN DEL SET COVERING PROBLEM

*Juan Salas[1], Mauricio Mora[1], Hugo Barriga[1], José-Miguel Rubio[1], Broderick Crawford[2]
[1] Universidad Tecnológica de Chile INACAP
[2] Pontificia Universidad Católica de Valparaíso
{jsalasf,mmorad,hbarriga,jrubiol}@inacap.cl
{broderick.crawford}@pucv.cl

RESUMEN

La presente investigación: Estudio de variación poblacional usando Harmony Search (HS) para la resolución del Set Covering Problem (SCP), logró demostrar los efectos del tamaño de la población de soluciones en la obtención de resultados cercanos al optimo global, lo cual es importante debido a que las Metaheurísticas más usadas como: Artificial Bee Colony (ABC), Black Hole (BH), Soccer league competition (SLC) y Gravitacional Search Algorithm (GSA) manejan población de tamaño fijo. Específicamente se compararon los resultados de implementación de dos tipos de resolución del SCP. para lo cual se utilizó una variación de la metaheurística HS. La primera forma de resolución empleó un tamaño de Población Fija (PF), mientras que la segunda usó Población Variable (PV) en la ejecución del algoritmo. Los resultados de ambas estrategias fueron comparados dando como resultado que la opción de población variable es la mejor para la mayoría de las instancias del benchmark.

Palabras Claves: Metaheurística, Población fija, Población Variable, Set Covering Problem.

ABSTRACT

The present study called: Study of the population variation using Harmony Search (HS) for the resolution of the Set Covering Problem (SCP), was able to demonstrate the effects of population size, in obtaining results close to the global optimum, which is important because most common metaheuristics like Artificial Bee Colony (ABC), Black Hole (BH), Soccer league competition (SLC) and Gravitacional Search Algorithm (GSA) make use of a population of fixed size. Specifically we compared the results of implementation of two types of resolution of the SCP for which a variation of the metaheuristic HS was used. The first form of resolution used a population of fixed size, while the second form of resolution varied the population throughout the execution of the algorithm. The results of both strategies were compared with the result that the variable population option is the best for most benchmark instances.

Keywords Metaheuristic, Fixed population, Variable population, Set Covering Problem.

* ***Autor Correspondiente:*** *Juan Salas F. Universidad Tecnológica de Chile INACAP. Chile*
E-mail: jsalasf@inacap.cl
Fecha de recepción: septiembre 2017; Fecha Aceptación: XXX 2017

INTRODUCCIÓN

La gran mayoría de las empresas se enfrenta a problemas de optimización diariamente (Talbi, 2009), lo cual justifica en gran medida investigaciones como la presente. Muchos problemas de las ciencias como de las empresas pueden ser modelados como problemas de optimización. Las metaheurísticas (MH) son algoritmos de optimización en las ciencias de la computación y matemáticas aplicadas, su nombre proviene del griego "meta" ("más allá", aquí con el sentido de "nivel superior") y "heurístico" (de ευρισκειν, heuriskein, "encontrar"). Generalmente se aplican a problemas que no tienen un algoritmo o heurística específica que dé una solución satisfactoria; o bien cuando no es posible implementar una técnica completa dado su alto costo computacional. Este tipo de algoritmos que están bajo el alero de la Inteligencia Artificial (IA) están generando gran interés en diversas tecnologías, industrias y servicios dado que han probado ser eficientes a la hora de resolver una gran cantidad de problemas de optimización de diferentes dominios en la vida real (Torres-Jiménez & Pavón, 2014). La instanciación de una MH requiere escoger entre un conjunto de diferentes parámetros y asignarle valores a cada uno de ellos, la cual es crucial para resolver las instancias del SCP (Birattari, Stützle, Paquete, & Varrentrapp, 2002). El SCP toma importancia en la práctica, dado que ha sido usado para modelar problemas complejos como Asignación y ruteo de trenes (Samà, DAriano, Toli, Pacciarelli & Corman, 2015), asignación de vehículos (Prata, 2016), control de acceso basado en roles (Chen & Crampton, 2009) entre otros. En este estudio se utilizará una variación de la MH HS en el espacio binario para resolver el SCP. Adicionalmente se comparará el comportamiento de la MH utilizando PF contra PV. Cuando hablamos de PF, corresponde a la cantidad de soluciones en el espacio de memoria, no cambia durante la ejecución del algoritmo, por el contrario, cuando hablamos de una PV, se agregan soluciones al espacio en memoria. Ambos métodos serán probados contra las instancias de la librería OR (Beasley, 1990). La meta principal consiste en determinar la mejor estrategia para solucionar el SCP. El trabajo se formula de la siguiente manera: En la sección Introducción se realiza una descripción de alto nivel de la técnica de inteligencia artificial a usar y se presenta brevemente la estructura de trabajo que dará origen a los resultados con miras a hacer comparativas y conclusiones. Posteriormente en la sección Materiales y Métodos se aborda la MH HS original, pasando por una pequeña muestra de algunas variaciones presentes en otros estudios, adicionalmente se propone una nueva variación de HS con la cual se resolverá el SCP. De igual modo se describe formalmente en términos matemáticos el problema a resolver. En la sección Resultados se despliegan los valores obtenidos por la MH, los cuales son analizados en profundidad para pasar a la sección Discusión.

MATERIALES Y MÉTODOS

El SCP es un problema matemático conocido, el cual intenta cubrir un conjunto de necesidades al menor costo posible. El SCP fue incluido en la lista de 21 problemas NP-*completo* de Karp (Karp, 2010). Existen muchos usos prácticos para este problema como lo son: planificación de producción en la industria (Vasko, Wolf, & Stott, 1987, 1989; Vasko, Wolf, Stott, & Scheirer, 1989), ruteo de vehículos (Balinski & Quandt, 1964; Foster & Ryan, 1976), planificación naviera (Fisher & Rosenwein, 1989; Bellmore, Geenberg, & Jarvis, 1970), ataques o defensas de redes (Bellmore & Ratliff, 1971), balanceo de líneas de ensamblaje (Freeman & Jucker, 1967; Salveson, 1955), asignación de tráfico en sistemas de comunicación satelitales (Ribeiro, Minoux, & Penna, 1989; Ceria, Nobili, & Sassano, 1998), simplificación de expresiones booleanas (Breuer, 1970), cálculo de límites en programación lineal (Christofides, 1971), obtención de información (Day, 1965), división distrital (Garfinkel & Nemhauser, 1970), asignación de tripulantes en líneas aéreas (Housos & Elmroth, 1997), entre otros.

El SCP puede ser formulado como sigue:

$$Minimizar(Z) = \sum_{j=1}^{n} c_j x_j \tag{1}$$

Sujeto a:

$$\sum_{j=1}^{n} a_{ij} x_j \geq 1 \quad \forall i \in I \tag{2}$$

$$x_j \in \{0,1\} \quad \forall j \in J \tag{3}$$

Sea $A = (a_{ij})$ una matriz $m \times n$ {0-1} donde $I = \{1,...,m\}$ y $J = \{1,...,n\}$ son las filas y columnas respectivamente. Formalmente podemos decir que la columna j puede cubrir a la fila i si $a_{ij} = 1$. El vector c_j contiene valores no negativos que representan el costo de seleccionar la columna j y x_j es la variable de decisión, la cual puede tomar el valor 1 si la columna j es seleccionada o 0 si no. El objetivo consiste en encontrar el conjunto de costos mínimos $S \subseteq J$, tal que cada fila $i \in I$ sea cubierta por al menos una columna j.

HS es un algoritmo metaheurístico poblacional, inspirado en el proceso de búsqueda de un estado de armonía perfecto (AQH por sus siglas en ingles) en la improvisación musical. Fue propuesto por Z. W. Geem et al. en (Geem, Kim, & Loganathan, 2001). La idea principal de la esta metaheurística es mimetizar el proceso realizado por músicos cuando ellos intentan tocar una armonía de alta calidad estética. Con el fin de entender correctamente que es lo que se considera como una buena solución en esta metaheurística, primero debemos entender el significado de AQH (Aesthetic Quality of Harmony). La AQH en un instrumento esta esencialmente determinada por la percepción de la frecuencia del sonido, la calidad del sonido y el volumen. La calidad del sonido es principalmente determinada por la forma de

las ondas o la modulación de la señal del sonido. Las armonías que pueden ser generadas dependen directamente del tono o rango de frecuencia del instrumento en particular (Geem, 2009). Diferentes notas tienen diferentes frecuencias, por ejemplo, la nota LA tiene la frecuencia fundamental 440*Hz*. La frecuencia fundamental de cada nota puede ser revisada en (Tabla 1). En el proceso de improvisación musical, todos los músicos ejecutan acordes en posible rango al mismo tiempo, para generar una armonía, si todos los acordes generan una buena armonía, cada músico almacena en su memoria esa experiencia y posibilita la ejecución de una buena armonía en la siguiente improvisación. Ocurre exactamente lo mismo en la optimización: La solución inicial es generada aleatoriamente con las variables de decisión en el rango predefinido. Si al evaluar la solución en la función objetivo, se obtiene un buen valor (buen fitness), quiere decir que hay una buena oportunidad de mejorar en la próxima iteración. La función objetivo f permite definir una relación de orden total entre cualquier par de soluciones del espacio de búsqueda. El óptimo global es definido como una solución $s^* \in S$ la cual posee el mejor fitness al ser evaluada en la función objetivo que cualquier otra solución del espacio de búsqueda, i.e., $\forall s \in S$, $f(s^*) \leq f(s)$.

Nota Musical	**Frecuencia**
Do	261,625565 Hz
Re	293,664768 Hz
Mi	329,627557 Hz
Fa	349,228231 Hz
Sol	391,995436 Hz
La	440,000000 Hz
Si	493,883301 Hz

Tabla 1. HS - Notas Musicales

La MH HS, consta de 5 pasos en su versión original (Zou, Gao, Li, & Wu, 2011). Dichos pasos serán revisados en detalle a continuación:

Paso 1: Inicialización del problema y los parámetros del algoritmo. Los parámetros x_{iL} y x_{iU} corresponden a los valores inferiores y superiores de las variables de decisión, esto aplica cuando HS es usada en dominio $\mathbb{R}$. Los parámetros requeridos para solucionar el problema de optimización son los siguientes: Harmony Memory Size (*HMS)* que corresponde al número de vectores de soluciones en el espacio de memoria, Harmony Memory Consideration Rate (*HMCR*) el cual determina el ratio de selección de valores desde memoria, Pitch Adjusting Rate (*PAR*) la cual determina la probabilidad de mejoras locales y finalmente el número de improvisaciones (*NI*).

Paso 2: Inicialización del espacio de memoria Harmony Memory (HM) hasta completar el *HMS*. Las armonías se improvisan de acuerdo a la ecuación 4.

$$x^{d}_{i}(t + 1) = x^{d}_{iL}(t) + rand()(x^{d}_{iU}(t) - x_{iL}{}^{d}(t)), \forall i \in \{1,2,...,n\}. \quad (4)$$

donde:

t = Iteración o improvisación actual.

d = Dimensión.

i = *i*-ésima solución o armonía.

$x^{d}_{i}(t + 1)$ = Representa a la siguiente solución.

$x^{d}_{iL}(t)$ = Valor inferior de la variable de decisión, dado *d* y *i*.

$x^{d}_{iU}(t)$ = Valor superior de la variable de decisión, dado *d* y *i*.

rand() = Corresponde a un valor aleatorio de una distribución uniforme dentro del rango [0,1].

Paso 3: La improvisación de una nueva armonía x'_i se logra mediante un proceso se puede ver el detalle en Algoritmo 1. De acuerdo al proceso, se generan vectores del tipo $(x^1, \dots, x^{HMS})$. Estos vectores componen una matriz del tipo que se muestra en la Figura 1:

$$HM = \begin{bmatrix} x_1^1 & \dots & x_n^1 \\ \vdots & \ddots & \vdots \\ x_1^{HMS} & \dots & x_n^{HMS} \end{bmatrix}$$

Figura 1. Matriz de espacio en memoria HM

Algoritmo 1: Generación de una nueva armonía

for i←1 **to** n **do**
  if $rand()$ ≤ HMCR **then**
    $x'_i = x_i^j \ (j = 1,2, \dots, HMS)$ //Consideración de uso de memoria.
    if $rand()$ ≤ PAR **then**
      $x'_i = x'_i \pm$ r(bw) // BW corresponde al ancho de banda
  Else
    $x'_i = x_{iL} + rand()(x_{iU} - x_{iL})$ //Selección Aleatoria

Algoritmo 1. Generación de una nueva armonía en HS tradicional

Paso 4: Actualización de *HM*. Si la nueva armonía generada $x' = (x'_1, x'_2, \dots, x'_n)$ tiene mejor fitness que la peor armonía x_{worst} en *HM*, entonces se reemplaza la peor armonía en *HM* con la nueva armonía tal como se muestra en el Algoritmo 2; de lo contrario se continua con el paso siguiente.

Algoritmo 2: Procedimiento para reemplazar peor armonía.

if $fitness(x') > fitness(x_{worst})$

$x_{worst} = x'$

Algoritmo 2. Procedimiento para reemplazar peor armonía.

Paso 5: Si el criterio de detención (e.g. máximo número de Improvisaciones *NI*) se ha alcanzado, entonces la ejecución del algoritmo finaliza. Por el contrario, se continua la ejecución a partir del paso 3.

Con el propósito de mejorar el desempeño en la cobertura de HS tradicional y mejorar ciertas deficiencias de esta MH, se presenta Global-Best Harmony Search (GHS), fue desarrollada por (Omran & Mahdavi, 2008). Una de las características de esta variación de HS, es la generación de la población inicial aleatoriamente usando un proceso de Bernoulli. Adicionalmente a ello, se genera un vector solución x_0 de manera voraz, basado en la idea de una relación de ganancia (profit ratio), donde se verifica si x_0 es mejor que x_{worst} en HM entonces x_{worst} es reemplazado por x_0. La metaheurística GHS ajusta dinámicamente el parámetro PAR de acuerdo a la ecuación (5):

$$PAR(t) = PAR_{min} - \frac{PAR_{max} - PAR_{min}}{NI} t \tag{5}$$

donde:

$PAR(t)$ = Pitch adjustment rate para la iteración t.

PAR_{min} = Valor mínimo que puede tomar *PAR*.

PAR_{max} = Valor máximo que puede tomar *PAR*.

NI = Número de improvisaciones.

t = Improvisación o iteración actual.

HS tradicional es buena explorando el espacio de búsqueda, pero no lo es tanto a la hora de explotar. Esto queda demostrado en (Xiang, An, Li, He, & Zhang, 2014). No existe un equilibrio entre la exploración y la explotación. HS fue pensada para ser utilizada en un espacio de búsqueda continua y no puede ser aplicada para resolver problemas de optimización combinatoria discretos. Para mejorar estos y otros aspectos se propone la variación de HS denominada Binary Global-Best Harmony Search (BGBHS). BGBHS, hace uso de los siguientes operadores: Pitch Adjustment Rate (PAR), Harmony Memory Consideration Rate (HMCR), reparación ADD y DROP, mecanismo de selección elitista y ajuste dinámico del parámetro *p* de Bernoulli entre otros.

El parámetro HMCR con valores altos, es útil a la hora de acelerar la velocidad de convergencia, mientras que con valores bajos permite escapar de óptimos locales, lo cual puede ser de gran utilidad hacia el fin de la ejecución. La variación del parámetro, puede ser revisado en la ecuación 6.

$$HMCR(t) = HMCR_{min} - \frac{HMCR_{max} - HMCR_{min}}{NI} t \tag{6}$$

donde:

$HMCR(t)$ = Harmony Memory Consideration Rate para la iteración t.

$HMCR_{min}$ = Mínimo valor que puede tomar $HMCR$.

$HMCR_{\max}$ = Máximo valor que puede tomar HMCR.

NI = Número de improvisaciones o iteraciones.

t = Iteración o improvisación actual.

Con el fin de mejorar la exploración en BGBHS, se propone una variación dinámica del parámetro de probabilidad p de Bernoulli al generar la población incial. El parámetro p parte con el valor 0,5 en la iteración o improvisación $t = 1$ y decrece con cada iteración, tendiendo a cero. El comportamiento del parámetro se ajusta a la ecuación (7):

$$p(t) = \frac{1}{t+1} \tag{7}$$

El grafico de la ecuación 7 puede ser apreciado en la Figura 2. En ella se puede comprobar como a medida que transcurren las iteraciones, el parámetro p tiende a cero.

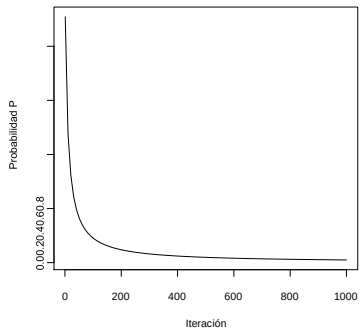

Figura 2. Valor de p a medida que aumentan las iteraciones

Este procedimiento permite que las soluciones al inicio de la generación tengan una mayor cantidad de variables activadas, debido a la alta probabilidad de éxito en el ensayo de Bernoulli. A medida que las iteraciones transcurran, el valor de p disminuye, lo cual produce que las soluciones tengan mucho menos variables activadas. Usando este simple método se asegura que el espacio de memoria HM, tenga un amplio espectro de soluciones desde el inicio, lo cual claramente contribuye a una mejor exploración del espacio de búsqueda.

En probabilidad y estadística, un proceso de Bernoulli es la repetición de un ensayo de Bernoulli. El ensayo consta de una secuencia finita o infinita de variables aleatorias binarias independientes x_1, x_2, x_3,..., $x_n \forall x_i \in \{0,1\}$ e $i = \{1,...,n\}$, los procesos de Bernoulli son un caso concreto de proceso estocástico

de tiempo discreto. El proceso de Bernoulli solo puede tomar dos valores, Éxito (ecuación 8) o Fracaso (ecuación 9), dada la probabilidad *p*.

$$P(x_i = 1) = P(\text{Éxito en el } i\text{-ésimo ensayo}) = p \quad (8)$$

$$P(x_i = 0) = P(\text{Fallo en el } i\text{-ésimo ensayo}) = 1 - p \quad (9)$$

Se generan tantas armonías como se defina en el parámetro *HMS*. Todas las armonías deben ser reparadas, dado que el proceso de Bernoulli no asegura la factibilidad de las soluciones generadas. El primero caso de reparación es cuando el vector generado viola las restricciones de la matriz A. El segundo caso es cuando el vector tiene activadas variables de alto costo, las cuales pueden ser reemplazadas por variables de menores costos manteniendo la factibilidad de la solución. En base a lo anterior, el operador de reparación consiste de dos fases: la primera llamada *ADD*, donde se procede a activar columnas del vector de manera tal que este se vuelva factible; la segunda se denomina *DROP* y se aplica para eliminar columnas redundantes sin que ello afecte la factibilidad de la armonía. El detalle de este operador puede ser revisado en el Algoritmo 3.

Se agrega un comportamiento elitista al funcionamiento de la MH, tratando de mejorar en cada iteración la mejor solución x_{best}, o bien mejorando la peor solución x_{worst} existente en HM. Un detalle de este comportamiento puede ser revisado en el algoritmo 4.

Algoritmo 3: Operador de reparación ADD y DROP.

```
//FASE ADD
M ← 1,2, … , m
Ai ∑_{j=1}^{n} c_j x_j  a_ij x_j, i ∈ M
for j←1 to n do
    if x_j = 0 and ∃i ∈ M, A_i < 1 then
        x_j ← 1
        A_i ← A_i + a_ij
    endif
endfor
//FASE DROP
for j←1 to n do
    if x_j = 1 and ∃i ∈ M, A_i - a_ij ≥ 1 then
        x_j ← 0
        A_i ← A_i − a_ij
    endif
endfor
```

Algoritmo 3. Operador de reparación ADD y DROP

Algoritmo 4: Mecanismo de selección elitista.

if $fitness(x') > fitness(x_{best})$

$x_{best} = x'$

if $fitness(x') > fitness(x_{worst})$

$x_{worst} = x'$

Algoritmo 4. Mecanismo de selección elitista.

RESULTADOS

Con el fin de probar la correcta ejecución de la MH, se realizó un total de 30 ejecuciones independientes de cada instancia del benchmark, tanto para PF como PV. El tiempo de ejecución de la MH contra cada uno del set de datos varió entre 2 y 120 minutos dependiendo del nivel de complejidad. La MH fue programada en lenguaje Python 2.7, haciendo uso de las librerías Numpy y Scipy. Como persistencia se utilizó archivos de texto y como procesamiento se contó con una máquina virtual con CPU doble núcleo 2.0 GHz (64 bit), 8GB de RAM, 80 GB de Disco Duro y sistema operativo Windows 7 Profesional de 64 bit. En la Tabla 2, la columna RPD reporta la desviación porcentual relativa entre el menor valor obtenido experimentalmente Z_{MIN} (Menor valor obtenido por MH) y el óptimo global para esa instancia Z_{BKS}. El RPD se calcula mediante la ecuación 10.

$$RPD = \frac{100(Z_{MIN} - Z_{BKS})}{Z_{BKS}} \quad (10)$$

La columna Z_{AVG} representa el promedio de las 30 ejecuciones para el benchmark en cuestión y la columna $\Delta\, RPD$ representa la diferencia entre el RPD de la PV y el RPD de la PF. Si $\Delta\, RPD > 0$ para una instancia dada, eso quiere decir que la PV fue más efectiva en la obtención de resultados.

		POBLACION FIJA				**POBLACION VARIABLE**				
Instancia	Z_{BKS}	Z_{MIN}	Z_{MAX}	Z_{AVG}	RPD	Z_{MIN}	Z_{MAX}	Z_{AVG}	RPD	$\Delta\, RPD$
4.1	429	468	505	483.30	9.09	456	488	473.60	6.29	2.80
4.2	512	611	675	635.03	19.34	583	675	612.77	13.87	5.47
4.3	516	587	643	611.60	13.76	565	637	586.30	9.50	4.26
4.4	494	569	623	598.53	15.18	548	615	579.67	10.93	4.25
4.5	512	581	646	621.03	13.48	552	643	593.47	7.81	5.66
4.6	560	648	712	673.70	15.71	624	685	649.10	11.43	4.29
4.7	430	495	552	516.83	15.12	475	541	502.73	10.47	4.65
4.8	492	560	607	584.13	13.82	526	588	557.87	6.91	6.91
4.9	641	775	877	817.87	20.90	748	810	777.53	16.69	4.21
4.10	514	596	648	616.03	15.95	580	624	473.60	12.84	3.11
5.1	253	291	317	302.93	15.02	289	314	296.43	14.23	0.79
5.2	302	355	388	370.93	17.55	341	389	365.03	12.91	4.64
5.3	226	257	272	264.77	13.72	253	272	260.47	11.95	1.77
5.4	242	266	291	276.53	9.92	264	283	271.03	9.09	0.83

5.5	211	246	265	254.07	16.59	240	258	249.30	13.74	2.84
5.6	213	264	286	275.27	23.94	253	295	269.23	18.78	5.16
5.7	293	342	376	353.13	16.72	334	365	347.77	13.99	2.73
5.8*	288	331	364	350.03	14.93	335	371	347.47	16.32	-1.39
5.9	279	330	359	341.90	18.28	325	360	338.43	16.49	1.79
5.10	265	294	314	304.47	10.94	290	317	301.07	9.43	1.51
6.1	138	163	179	170.50	18.12	155	168	161.20	12.32	5.80
6.2	146	170	190	179.90	16.44	164	184	172.10	12.33	4.11
6.3	145	169	189	179.10	16.55	165	179	171.37	13.79	2.76
6.4	131	148	160	155.17	12.98	142	155	149.27	8.40	4.58
6.5	161	193	212	201.27	19.88	186	205	195.73	15.53	4.35
A.1	253	283	307	293.40	11.86	280	304	290.53	10.67	1.19
A.2	252	294	324	312.03	16.67	291	324	309.30	15.48	1.19
A.3	232	274	295	284.37	18.10	268	288	280.13	15.52	2.59
A.4	234	269	294	282.07	14.96	267	287	277.33	14.10	0.85
A.5	236	271	292	281.87	14.83	261	293	278.63	10.59	4.24
B.1	69	85	93	89.47	23.19	82	92	88.77	18.84	4.35
B.2	76	92	105	95.20	21.05	90	97	92.90	18.42	2.63
B.3	80	89	95	91.70	11.25	87	94	90.13	8.75	2.50
B.4	79	92	104	96.63	16.46	91	102	95.07	15.19	1.27
B.5*	72	82	94	88.53	13.89	86	91	88.20	19.44	-5.56
C.1	227	261	275	267.68	14.98	256	279	265.50	12.78	2.20
C.2	219	260	280	268.82	18.72	256	275	265.50	16.89	1.83
C.3**	243	298	342	310.71	22.63	298	318	307.92	22.63	0.00
C.4	219	264	287	274.82	20.55	261	290	275.21	19.18	1.37
C.5**	215	244	290	257.68	13.49	244	273	256.40	13.49	0.00
D.1**	60	66	76	71.37	10.00	66	87	71.80	10.00	0.00
D.2**	66	76	86	79.52	15.15	76	88	79.50	15.15	0.00
D.3	72	82	93	85.48	13.89	81	104	86.29	12.50	1.39
D.4	62	70	83	72.85	12.90	69	96	73.54	11.29	1.61
D.5*	61	69	80	71.71	13.11	70	76	71.88	14.75	-1.64

Nota: Los valores denotados con * indican instancias donde PF ganó a PV. Lo valores denotados con ** indican que tanto PF como PV llegaron al mismo resultado. En resto de instancias ganó PV.

Tabla 2. Resumen de resultados por instancia PF y PV.

Con el fin de poder demostrar de manera correcta la regularidad y la consistencia de los datos, se utilizó el modelo de análisis estadístico propuesto en (José Lanza, 2014).

En este caso se analizó las dos versiones del algoritmo PF y PV. Siguiendo el modelo referido, la primera acción consistió en determinar la existencia de outliers en los resultados de diferentes instancias, donde se evaluó mediante diferentes técnicas la existencia o no de estos elementos. Ejemplo de ellos es

la figura 3, donde mediante una gráfica de cajas se determina la existencia de outliers para la PF dentro de los resultados de la instancia SCP41.

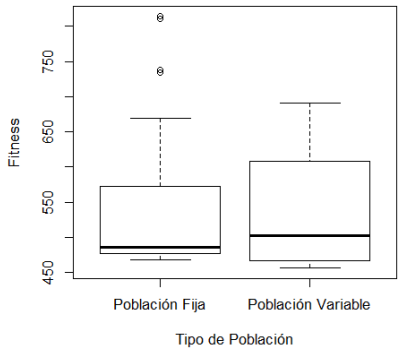

Figura 3. Boxplot instancia SCP41 para la detección de outliers.

Se verificó la normalidad de los datos mediante los test Shapiro-Wilk y Kolmogorov-Smirnov-Lilliefors, para lo cual se plantearon las siguientes hipótesis:

$$H_0 = Los\ datos\ siguen\ una\ distribución\ normal$$

$$H_1 = Los\ datos\ no\ siguen\ una\ distribución\ normal$$

Dado los p-valores obtenidos en los test, se rechaza H_0. Esto permitió aplicar Wilcoxon-Mann-Whitney para verificar superioridad de la estrategia de resolución con PV sobre PF. Se define las hipótesis:

$$H_0 = \bar{X}\ Algoritmo\ con\ población\ fija \leq\ \bar{X}\ Algoritmo\ con\ población\ variable$$

$$H_1 = \bar{X}\ Algoritmo\ con\ población\ fija >\ \bar{X}\ Algoritmo\ con\ población\ variable$$

Se ejecutó el análisis con el programa estadístico R, con el cual se obtuvo un p-valor < 0.05 motivo por el cual se rechaza H_0 y se acepta H_1 esto implica que PV proporciona mejores resultados.

DISCUSIÓN Y CONCLUSIÓN

Los resultados comparados directamente a través de $\Delta\ RPD$ muestran que, en la mayoría de las instancias probadas, la versión de la MH BGBHS con PV es mejor. Sin embargo, existen instancias en las cuales no se mostró mejorías denotadas con * o bien ambas técnicas llegaron al mismo resultado ** en la Tabla 2.

Consistentemente, el análisis estadístico realizado demuestra que el comportamiento de la MH con PV obtiene mejores resultados en 84.4% de las instancias. Llama la atención el hecho de que la variación de población encuentre mejores resultados incluso a menor cantidad de iteraciones lo cual es

bastante prometedor debido a que disminuye el tiempo de procesamiento computacional, con lo cual muchos problemas pueden ser resueltos en una menor cantidad de tiempo.

En el presente trabajo se realizó una variación positiva en el tamaño de la población (Agregando soluciones al espacio en memoria hasta un máximo predefinido), pero no se profundizó en realizar un ajuste dinámico, en el sentido de agregar o quitar soluciones del espacio de memoria. Esto abre claramente la opción para investigar la posibilidad de utilizar una población completamente adaptativa.

Si miramos la Figura 4, muestra estancamiento en óptimos globales, claramente hacia el final la tendencia es asintótica a x. Por el contrario, la Figura 5 presenta un comportamiento siempre decreciente, sin estancamientos en óptimos locales, mostrando una gran capacidad de exploración, con tendencia al óptimo global.

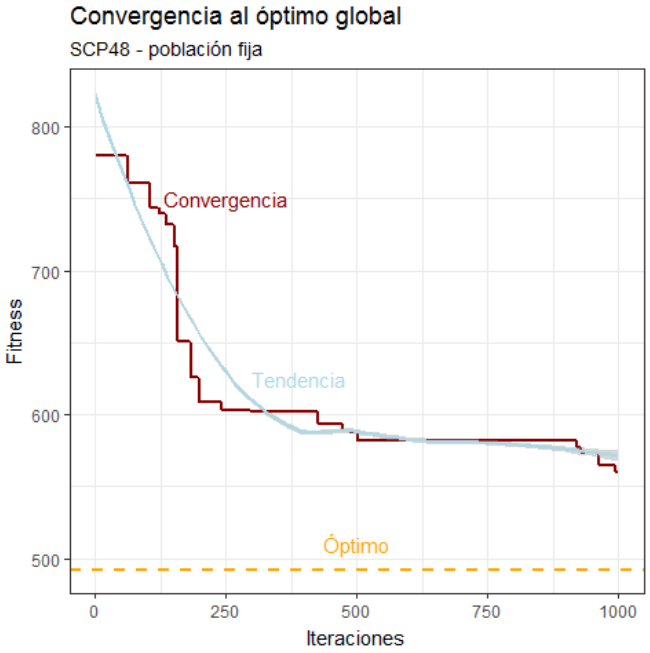

Figura 4. Grafica de convergencia SCP48, obteniendo como mejor solución 468.

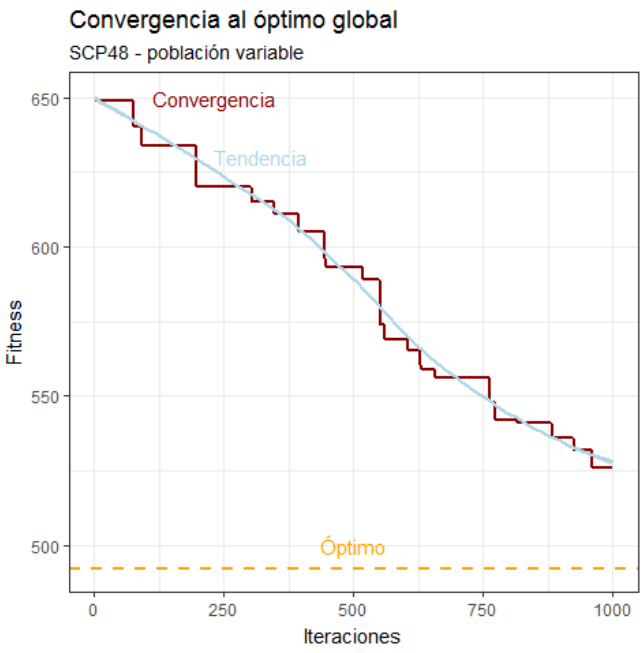

Figura 5. Grafica de convergencia SCP54, obteniendo como mejor solución 456.

Finalmente queda demostrado que para resolver problemas de optimización como SCP, es útil emplear PV en contraposición a PF, con el fin de lograr mejores resultados.

REFERENCIAS

Balinski, M. L., & Quandt, R. E. (1964). On an integer program for a delivery problem.

Operations Research, *12*(2), 300–304. doi: 10.1287/opre.12.2.300

Beasley, J. E. (1990). OR-library: distributing test problems by electronic mail. *Journal of the Operational Research Society*, *41*(11), 1069–1072.

Beheshti, Z., & Shamsuddin, S. M. H. (2013). A review of population-based meta-heuristic algorithm. International Journal of Avances in Soft Computing and its Applications, 5(1).

Bellmore, M., Geenberg, H. J., & Jarvis, J. J. (1970). Multi-commodity disconnecting sets. *Management Science*, *16*(6), B427–B433.

Bellmore, M., & Ratliff, H. D. (1971). Optimal defense of multi-commodity networks. *Management Science*, *18*(4-part-i), B174–B185. doi: 10.1287/mnsc.18.4.B174

Birattari, M., Stützle, T., Paquete, L., & Varrentrapp, K. (2002). A racing Algoritmo for configuring metaheuristics. In W. B. Langdon et al. (Eds.), *GECCO 2002: Proceedings of the genetic and evolutionary computation conference, new york, usa, 9-13 july 2002* (pp. 11–18). Morgan Kaufmann.

Breuer, M. A. (1970, January). Simplification of the covering problem with application to boolean expressions. *Journal of the Association for Computing Machinery*, *17*(1), 166–181. doi: 10.1145/321556.321572

Ceria, S., Nobili, P., & Sassano, A. (1998). A lagrangian-based heuristic for large-scale set covering problems. *Mathematical Programming*, *81*(2), 215–228. doi: 10.1007/BF01581106

Chen, L., & Crampton, J. (2009). Set covering problems in role-based access control. In *ESORICS* (Vol. 5789, pp. 689–704). Springer.

Christofides, N. (1971). Zero-one programming using non-binary tree-search. *Computer Journal*, *14*(4), 418–421.

Day, R. H. (1965). Letter to the editor on optimal extracting from a multiple file data storage system: An application of integer programming. *Operations Research*, *13*(3), 482–494. doi: 10.1287/opre.13.3.482

Ferreira, M., Bagarić, J., Lanza-Gutierrez, J. M., Priem-Mendes, S., Pereira, J. S., & Gomez-Pulido, J. A. (2015). On the use of perfect sequences and genetic algorithms for estimating the indoor location of wireless sensors. International Journal of Distributed Sensor Networks, 11(4), 720574.

Fisher, M. L., & Rosenwein, M. B. (1989). An interactive optimization system for bulk-cargo ship scheduling. *Naval Research Logistics*, *36*(1), 27–42. doi: 10.1002/1520-6750(198902)36:1¡27::AIDNAV3220360103¿3.0.CO;2-0

Foster, B. A., & Ryan, D. M. (1976). An integer programming approach to the vehicle scheduling

problem. *Operations Research*, *27*, 367–384.

Freeman, B. A., & Jucker, J. V. (1967). The line balancing problem. *Journal of Industrial Engineering*, *18*, 361–364.

Garfinkel, R. S., & Nemhauser, G. L. (1970). Optimal political districting by implicit enumeration techniques. *Management Science*, *16*(8), B495–B508. doi: 10.1287/mnsc.16.8.B495

Geem, Z. W. (2009). Music-inspired harmony search Algoritmo: theory and applications (Vol. 191). Springer.

Geem, Z. W., Kim, J., & Loganathan, G. V. (2001). A new heuristic optimization Algoritmo: Harmony search. *Simulation*, *76*(2), 60–68. Retrieved from http://dx.doi.org/10.1177/003754970107600201 doi: 10.1177/003754970107600201

Housos, E., & Elmroth, T. (1997). Automatic optimization of subproblems in scheduling airline crews. *Interfaces*, *27*(5), 68-77. doi: 10.1287/inte.27.5.68

Karp, R. M. (2010). Reducibility among combinatorial problems. In M. Jünger et al. (Eds.), *50 years of integer programming 1958-2008 - from the early years to the state-of-the-art* (pp. 219–241). Springer. Retrieved from http://dx.doi.org/10.1007/978-3-540-68279-0˙8 doi: 10.1007/978-3-540-68279-0˙8

Jose Lanza, Assuming multiobjective metaheuristics to solve a three-objective optimisation problem for relay node deployment in wireless sensor networks, Applied Soft Computing, (2014), pp. 675–687.

Omran, M. G. H., & Mahdavi, M. (2008). Global-best harmony search. *Applied Mathematics and Computation*, *198*(2), 643–656. Retrieved from http://dx.doi.org/10.1016/j.amc.2007.09.004 doi: 10.1016/j.amc.2007.09.004

Prata, Bruno de Athayde. (2016). A multiobjective metaheuristic approach for the integrated vehicle and crew scheduling. Journal of Transport Literature, 10(2), 10-14. https://dx.doi.org/10.1590/2238-1031.jtl.v10n2a2

Ribeiro, C. C., Minoux, M., & Penna, M. C. (1989). An optimal column-generation-with-ranking Algoritmo for very large scale set partitioning problems in traffic assignment. *European Journal of Operational Research*, *41*(2), 232–239.

Salveson, M. E. (1955). The assembly line balancing problem. *Journal of Industrial Engineering*, *6*, 18–25.

Samà, M., DAriano, A., Toli, A., Pacciarelli, D., & Corman, F. (2015, September). Metaheuristics for

real-time near-optimal train scheduling and routing. In Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on (pp. 1678-1683). IEEE.

Talbi, E. (2009). Metaheuristics - from design to implementation. Wiley.

Torres-Jiménez, J., & Pavón, J. (2014). Applications of metaheuristics in real-life problems. *Progress in AI*, *2*(4), 175–176. Retrieved from https://doi.org/10.1007/s13748-014-0051-8 doi: 10.1007/s13748-0140051-8

Vasko, F. J., Wolf, F. E., & Stott, K. L. (1987, June). Optimal selection of ingot sizes via set covering. *Operations Research*, *35*(3), 346-353.

Vasko, F. J., Wolf, F. E., & Stott, K. L. (1989). A set covering approach to metallurgical grade assignment. *European Journal of Operational Research*, *38*(1), 27–34.

Vasko, F. J., Wolf, F. E., Stott, K. L., & Scheirer, J. W. (1989). Selecting optimal ingot sizes for bethlehem steel. *Interfaces*, *19*(1), 68–84. doi: 10.1287/inte.19.1.68

Xiang, W., An, M., Li, Y., He, R., & Zhang, J. (2014). An improved global-best harmony search Algoritmo for faster optimization. *Expert Syst. Appl.*, *41*(13), 5788–5803. Retrieved from http://dx.doi.org/10.1016/j.eswa.2014.03.016 doi: 10.1016/j.eswa.2014.03.016

Zou, D., Gao, L., Li, S., & Wu, J. (2011). Solving 0-1 knapsack problem by a novel global harmony search Algoritmo. *Appl. Soft Comput.*, *11*(2), 1556–1564. Retrieved from http://dx.doi.org/10.1016/j.asoc.2010.07.019 doi: 10.1016/j.asoc.2010.07.019