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

\*Juan Salas1, Mauricio Mora1, Hugo Barriga1, José-Miguel Rubio1, Broderick Crawford2

1 Universidad Tecnológica de Chile INACAP

2 Pontificia Universidad Católica de Valparaíso

{jsalasf,mmorad,hbarriga,jrubiol}@inacap.cl

{broderick.crawford}@pucv.cl

# RESUMEN

El presente trabajo se titula: Estudio de variación poblacional usando Harmony Search (HS) para la resolución del Set Covering Problem (SCP). Su aporte principal consiste en demostrar los efectos del tamaño de la población de soluciones, en la obtención de resultados a un problema de optimización. Los efectos del presente estudio son relevantes, debido a que la mayoría de las metaheurísticas actuales usan población de soluciones de tamaño fijo. Nuestro equipo logró comprobar que una población de soluciones variable permite obtener mejores resultados. Este documento se estructura de la siguiente forma: En la sección introducción se realiza una revisión y justificación de este estudio; en la sección materiales y métodos se define el marco de trabajo de esta investigación; en la sección resultados se presentan los valores obtenidos de los diferentes experimentos y finalmente en la sección discusión y conclusión se muestran los comentarios más importantes respecto a los resultados obtenidos.

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

# ABSTRACT

The present work is entitled: Study of population variation using Harmony Search (HS) for the resolution of Set Covering Problem (SCP). Its main contribution is to demonstrate the effects of the population size of solutions, in obtaining results to an optimization problem. The effects of the present study are relevant, since most of the current metaheuristics use population of fixed size solutions. Our team was able to verify that a population of variable solutions allows to obtain better results. This document is structured as follows: In the introduction section, a review and justification of this study is carried out; in the materials and methods section, the framework of this research is defined; In the results section the values ​​obtained from the different experiments are presented and finally in the discussion and conclusion section the most important comments regarding the obtained results are shown.

**Keywords:** Metaheuristic, Harmony Search, Fixed size population, Variable population, Set Covering Problem.

# INTRODUCCIÓN

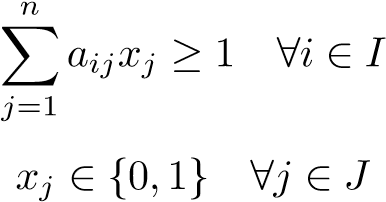
Las metaheurísticas (MH) son algoritmos de optimización muy utilizadas en las ciencias de la computación y matemáticas aplicadas, su nombre proviene del griego "meta" ("nivel superior") y "heurístico" ("encontrar"). Se aplican a problemas de optimización que no tienen algoritmo o heurística específica que dé una solución satisfactoria; o bien cuando no es posible implementar una técnica completa dado su alto costo computacional. Este tipo de algoritmos pertenecientes a la Inteligencia Artificial (IA) generan interés en diversas tecnologías, industrias y servicios dada su eficiencia en resolver problemas de optimización de diferentes dominios en la vida real (Torres-Jiménez & Pavón, 2014). La parametrización de la MH es crucial para resolver las instancias del Set Covering Problem (SCP) (Birattari et al., 2002). Dentro de los parámetros que recibe la MH, se encuentra el tamaño de la población. En este trabajo se identificó el efecto positivo de aumentar la cantidad de soluciones en el espacio de memoria en la ejecución del algoritmo. Se comparó el comportamiento de la MH utilizando Población de soluciones Fijas (PF) contra Población de soluciones Variables (PV). Ambos métodos fueron probados contra las instancias de la librería OR (Beasley, 1990). La meta principal consistió en determinar la mejor estrategia para solucionar el SCP ya sea usando PF o PV. El trabajo se estructura de la siguiente manera: En la sección Introducción se realiza una descripción de alto nivel de la técnica de inteligencia artificial usada y se presenta brevemente la estructura de trabajo que proporcionó los resultados que permiten 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 y Conclusiones.

# MATERIALES Y MÉTODOS

El SCP es un problema matemático ampliamente conocido, el cual intenta cubrir un conjunto de necesidades al menor costo posible cuando hablamos de minimización. El SCP fue incluido en la lista de 21 problemas NP-*completos* de Karp (Karp, 2010). Existen muchos usos prácticos para este problema como lo son: Localización de unidades Roadside en redes vehiculares (Sarubbi et al., 2017), ubicación de instalaciones (Crawford et al., 2017), ruteo de vehículos (Cacchiani, Hemmelmayr, & Tricoire, 2014; Prata, 2016; Yakici, 2017), asignación de tráfico en sistemas de comunicación satelitales (Ribeiro, Minoux, & Penna, 1989; Ceria, Nobili, & Sassano, 1998), asignación de tripulantes en líneas aéreas (Housos & Elmroth, 1997), asignación y ruteo de trenes (Samà et al., 2015), control de acceso basado en roles (Chen & Crampton, 2009) entre otros

El SCP puede ser formulado como sigue:

(1)

Sujeto a:

(2)

(3)

Sea una matriz donde y son las filas y columnas respectivamente. Formalmente podemos decir que la columna *j* puede cubrir a la fila *i* si *aij* = 1. El vector *cj* contiene valores no negativos que representan el costo de seleccionar la columna *j* y *xj* es la variable de decisión, la cual puede tomar el valor 1 si la columna *j* es seleccionada o 0 si no.

HS es un algoritmo metaheurístico del tipo poblacional (Beheshti & Shamsuddin, 2013) como son: Artificial Bee Colony (Crawford et al., 2014), Black Hole (Gómez et al., 2016) y Soccer League Competition (Jaramillo et al., 2016). Harmony Search (HS) está inspirada en el proceso de búsqueda de un estado de armonía perfecto o Aesthetic Quality of Harmony (AQH) en la improvisación musical. Fue propuesto por Z. W. Geem (Geem, Kim, & Loganathan, 2001). La idea de esta MH es mimetizar el proceso realizado por músicos cuando intentan tocar una armonía. Las armonías que pueden ser generadas dependen directamente del tono o rango de frecuencia del instrumento (Geem, 2009). Diferentes notas tienen diferentes frecuencias, por ejemplo, la nota LA tiene la frecuencia fundamental 440*Hz*. En el proceso de improvisación musical, todos los músicos ejecutan acordes en un 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, quiere decir que hay una buena oportunidad de mejorar en la próxima iteración. La función objetivo permite definir una relación de orden total entre cualquier par de soluciones del espacio de búsqueda. El óptimo global en minimización, es definido como una solución *s*∗ ∈ *S*, donde *S* corresponde al conjunto total de soluciones. *s*∗ posee el mejor fitness (menor valor) al ser evaluada en la función objetivo que cualquier otra solución del espacio de búsqueda, i.e., ∀*s* ∈ *S*, *f*(*s*∗) ≤ *f*(*s*). El fitness corresponde al valor obtenido al evaluar la solución en la función objetivo.

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

Paso 1: Inicialización del problema y los parámetros del algoritmo. Los parámetros *xiL* y *xiU* corresponden a los valores inferiores y superiores de los dominios de las variables de decisión, esto aplica cuando HS es usada en dominio . 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 HM hasta completar el *HMS*. Las armonías se improvisan de acuerdo a la ecuación (4).

(4)

donde:

*t* = Iteración o improvisación actual.

*d* = *i*-ésima variable del vector solución o armonía.

*i* = *i*-ésima solución o armonía

*xdi*(*t* + 1) = Representa a la siguiente solución.

*xdiL*(*t*) = Valor inferior de la variable de decisión, dado *d* y *i.*

*xdiU*(*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 se logra mediante un proceso se puede ver el detalle en Algoritmo 1. De acuerdo al proceso, se generan vectores del tipo . Estos vectores componen una matriz del tipo que se muestra en la Figura 1:

**Figura 1.** Matriz de espacio en memoria HM

|  |
| --- |
| **Algoritmo 1:** Generación de una nueva armonía |
| **for** i←1 **to** n **do**  **if** ≤ HMCR **then**  = //Consideración de uso de memoria.  **if** ≤ PAR **then**  = ± r(bw) // BW corresponde al ancho de banda  **Else**  = //Selección Aleatoria |

**Algoritmo 1.** Generación de una nueva armonía en HS tradicional

Paso 4: Actualización de *HM*. Si la nueva armonía generada tiene mejor fitness que la peor armonía 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** () > ()  = |

**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), 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 (Bertsekas & Tsitsiklis, 2002). Adicionalmente, se genera un vector solución de manera elitista, basado en la idea de una relación de ganancia (profit ratio), donde se verifica si es mejor que *xworst* en HM entonces *xworst* es reemplazado por . La metaheurística GHS ajusta dinámicamente el parámetro PAR de acuerdo a la ecuación (5):

(5)

donde:

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

*PARmin* = Valor mínimo que puede tomar *PAR*.

*PARmax* = 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 et al., 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).

(6)

donde:

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

*HMCRmin* = 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 (Salas et al., 2016), se propone una variación dinámica del parámetro de probabilidad *p* de Bernoulli al generar la población inicial. 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):

(7)

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.

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 o bien mejorando la peor solución 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 ←  Ai aijxj,i ∈ M  **for** j←1 **to** n **do**  **if** *xj* = 0 and ∃*i* ∈ *M,Ai <* 1 **then**      **endif**  **endfor**  //FASE DROP  **for** j←1 **to** n **do**  **if** *xj* = 1 and ∃*i* ∈ *M,Ai*  - ≥ 1 **then**      **endif**  **endfor** |

**Algoritmo 3.** Operador de reparación ADD y DROP

|  |
| --- |
| **Algoritmo 4:** Mecanismo de selección elitista. |
| **if** () < ()  =  **if** () < ()  = |

**Algoritmo 4.** Mecanismo de selección elitista.

La operatividad que permite agregar soluciones al HM logrando la implementación automática de PV, se basa en el proceso de generación inicial, tomando en cuenta un porcentaje de no mejora. Si transcurridas 100 iteraciones, se detecta estancamiento en óptimo local, se desencadena un proceso que se explica en algoritmo 5.

|  |
| --- |
| **Algoritmo 5:** Inclusión de nueva armonía en HM |
| nuevas\_armonias[] = inicialización de vector largo 10  **for** j←0 **to** 8 **do**  nueva\_armonia = nueva armonía con Proceso de Bernoulli usando  nueva\_armonia = reparacion(nueva\_armonia)  nuevas\_armonias[j]=nueva\_armonia  **endfor**  nuevas\_armonias[9] = nueva armonía elitista  mejor\_armonia = obtención de mejor armonía en base a fitness (nuevas\_armonias)  **if** fitness(mejor\_armonia) < fitness(x\_best)  = mejor\_armonia  **if** fitness(mejor\_armonia) < fitness(x\_worst)  = mejor\_armonia |

**Algoritmo 5.** Inclusión de nuevas armonías en HM

# 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 las tablas expuestas en esta sección, la columna RPD reporta la desviación porcentual relativa entre el menor valor obtenido experimentalmente y el óptimo global para esa instancia . El RPD se calcula mediante la ecuación (8).

(8)

La columna representa el promedio de las 30 ejecuciones para el benchmark en cuestión y la columna representa la diferencia entre el RPD de la PV y el RPD de la PF. Si para una instancia dada, eso quiere decir que la PV fue más efectiva en la obtención de resultados.

Tal como apreciamos, en la Tabla 1, los resultados obtenidos por PV para la instancia 4, fueron más cercanos al óptimo global, es decir: presentaron un mejor fitness.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **POBLACION FIJA** | | | | **POBLACION VARIABLE** | | | |  |
| Instancia |  |  |  |  |  |  |  |  |  |  |
| 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 |

**Tabla 1.** Resultados instancias 4.

En la Tabla 2, se puede ver como la estrategia PV obtiene mejores resultados para casi todos los sets de pruebas. Solo se ve superada por PF en el set 5.8. Las tablas 3 y 4 muestran como consistentemente la técnica con PV presentó mejor desempeño que la técnica con PF.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **POBLACION FIJA** | | | | **POBLACION VARIABLE** | | | |  |
| Instancia |  |  |  |  |  |  |  |  |  |  |
| 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 |

**Tabla 2.** Resultados instancia 5.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **POBLACION FIJA** | | | | **POBLACION VARIABLE** | | | |  |
| Instancia |  |  |  |  |  |  |  |  |  |  |
| 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 |

**Tabla 3.** Resultados instancia 6.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **POBLACION FIJA** | | | | **POBLACION VARIABLE** | | | |  |
| Instancia |  |  |  |  |  |  |  |  |  |  |
| 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 |

**Tabla 4.** Resultados instancia A.

La Tabla 5, muestra que la técnica PF solo fue mejor en uno de los conjuntos de datos de la instancia B.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **POBLACION FIJA** | | | | **POBLACION VARIABLE** | | | |  |
| Instancia |  |  |  |  |  |  |  |  |  |  |
| 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 |

**Tabla 5.** Resultados instancia B.

En las instancias superiores, C y D se presentan comportamientos donde ambas técnicas llegan a igualdad de soluciones, como es el caso de C.3, C.5, D.1 y D.2. Solamente en la instancia D.5 la técnica con PF es mejor. Para el resto, PV obtiene resultados más cercanos al óptimo global.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **POBLACION FIJA** | | | | **POBLACION VARIABLE** | | | |  |
| Instancia |  |  |  |  |  |  |  |  |  |  |
| 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 |

**Tabla 6.** Resultados instancia C.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **POBLACION FIJA** | | | | **POBLACION VARIABLE** | | | |  |
| Instancia |  |  |  |  |  |  |  |  |  |  |
| 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 |

**Tabla 7.** Resultados instancia D.

Con el fin de poder demostrar de manera correcta la regularidad y la consistencia de los datos, se utilizó el modelo de análisis estadístico presentado en el trabajo de José Lanza (Lanza-Gutierrez & Gomez-Pulido, 2015).

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 (observación estadística marcadamente diferente a otros valores de la muestra) 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 2, donde mediante una gráfica de cajas se determina la existencia de outliers para la PF dentro de los resultados de la instancia SCP41.



**Figura 2.** 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:

Dado los p-valores obtenidos en los test, se rechaza . Esto permitió aplicar Wilcoxon-Mann-Whitney (Mann & Whitney, 1947) para verificar superioridad de la estrategia de resolución con PV sobre PF. Se define las hipótesis:

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 y se acepta esto implica que PV proporciona mejores resultados.

# DISCUSIÓN Y CONCLUSIÓN

Los resultados comparados directamente a través de 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 \*\* como se ve 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 3, muestra estancamiento en óptimos locales, claramente hacia el final la tendencia es asintótica a . Por el contrario, la Figura 4 presenta un comportamiento siempre decreciente, sin estancamientos en óptimos locales, mostrando una gran capacidad de exploración, con tendencia al óptimo global.



|  |  |
| --- | --- |
| **Figura 3.** Gráfica de convergencia SCP48, obteniendo como mejor solución 560. | **Figura 4.** Gráfica de convergencia SCP48, obteniendo como mejor solución 526. |

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

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).

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.

Bertsekas, D. P., & Tsitsiklis, J. N. (2002). Introduction to probability (Vol. 1). Belmont, MA: Athena Scientific.

Cacchiani, V., Hemmelmayr, V. C., & Tricoire, F. (2014). A set-covering based heuristic algorithm for

the periodic vehicle routing problem. Discrete Applied Mathematics, 163(Pt 1), 53–64. http://doi.org/10.1016/j.dam.2012.08.032

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.

Crawford, B., Soto, R., Monfroy, E., Astorga, G., García, J., & Cortes, E. (2017, September). A Meta-

Optimization Approach for Covering Problems in Facility Location. In *Workshop on Engineering Applications* (pp. 565-578). Springer, Cham.

Crawford, B., Soto, R., Cuesta, R., & Paredes, F. (2014). Application of the artificial bee colony

algorithm for solving the set covering problem. The Scientific World Journal, 2014.

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

Gómez, A., Crawford, B., Soto, R., Jaramillo, A., Mansilla, S., Salas, J., & Olguín, E. (2016, June). An

binary black hole algorithm to solve the set covering problem. In Information Systems and Technologies (CISTI), 2016 11th Iberian Conference on (pp. 1-5). IEEE.

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

Jaramillo, A., Crawford, B., Soto, R., Villablanca, S. M., Rubio, Á. G., Salas, J., & Olguín, E. (2016,

August). Solving the set covering problem with the Soccer League Competition algorithm. In International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (pp. 884-891). Springer International Publishing.

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

Lanza-Gutierrez j., & Gomez-Pulido j. (2015). Assuming multiobjective metaheuristics to solve a three-

objective optimisation problem for Relay Node deployment in Wireless Sensor Networks. Applied Soft Computing, 30, 675-687.

Mann, H. B., & Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically

larger than the other. *The annals of mathematical statistics*, 50-60.

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.

Salas, J., Crawford, B., Soto, R., Rubio, Á. G., Jaramillo, A., Villablanca, S. M., & Olguín, E. (2016,

August). Binary Harmony Search Algorithm for Solving Set-Covering Problem. In International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (pp. 917-930). Springer International Publishing.

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.

Sarubbi, J. F., Silva, T. R., Martins, F. V., Wanner, E. F., & Silva, C. M. (2017, May). A GRASP based

heuristic for Deployment Roadside Units in VANETs. In *Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on* (pp. 369-376). IEEE.

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

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

Yakıcı, E. (2017). A heuristic approach for solving a rich min-max vehicle routing problem with mixed

fleet and mixed demand. *Computers & Industrial Engineering*, *109*, 288-294.

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