Algoritmos Evolutivos - Algoritmo Genético híbrido con búsqueda local

Hamilton Smith Gómez Osorio y Juan Carlos Rivera
hsgomezo@eafit.edu.co,jrivera6@eafit.edu.co

Departamento de Ciencias Matemáticas
Escuela de Ciencias
Universidad EAFIT
Medellín - Colombia

7 de diciembre de 2020

Resumen

Este trabajo presenta la construcción e implementación de un Método Evolutivo conocido como Algoritmo Genético híbrido, el cual integra diferentes desarrollos desde métodos constructivos, métodos constructivos aleatorizados y de búsqueda local, para encontrar una solución heurística a veinte instancias del Problema de la Mochila Multidimencional y Multiobjetivo, MO-MKP. Los resultados van desde la parametrización del modelo, mostrando la influencia de factores aleatorios en las soluciones obtenidas; la evaluación de tiempos de cómputo, factibilidad y aproximación de la Frontera de Pareto; como también el análisis respecto a diferentes algoritmos diseñados previamente.
Parlabras Claves: Problema de la Mochila, Algoritmos Genéticos híbridos, Búsqueda Local, Problemas multiobjetivo.

1. Introducción

En el mundo real existen múltiples problemas matemáticos que se consideran difíciles de resolver de manera análítica, en los cuales ni las herramientas computacionales permiten obtener la respuesta exacta. En los procesos de optimización, el tipo de problemas mencionados anteriormente son bastante comunes, lo cuales se conocon como NP, o problemas de tiempo polinomial no determinista,[8]. Para este tipo de problema se han desarrollado diferentes estrategias que utilizan métodos aproximados de contrucción y mejora de soluciones que se encuentren cerca del resultado óptimo, en caso de ser este el objetivo; o que por el contrario permiten utilizar soluciones iniciales de fácil construcción para otro tipo de procesos. Estos desarrollos se conocen como métodos heurísticos, los cuales no solo han mostrado ser de gran utilidad para la solución de problemas de gran dimensión, sino que se caracterizan por su corto tiempo de cómputo y el gran alcance de aplicabilidad que poseen.

Uno de los problemas considerados NP se conoce como Problema de la Mochila (Knapsack Problem), el cual hace referencia un problema de optimización combinatoria [2], que se caracteriza por la selección de diferentes objetos, los cuales llevan asignados unos pesos, donde se pretende cumplir con las restricciones de carga total maximizando las utilidades que estos elementos elegidos poseen. Este tipo de problema, cuando se está trabajando con un número de objetos candidados a ser elegidos, presenta una complejidad muy alta a la hora de ser resuelto por métodos exactos, dada la amplia cantidad de posibles combinaciones de elementos a seleccionar. Por lo que ha sido de gran interés por los investigadores y logrando así múltiples algoritmos heurísticos desarrollados para este problema. así, basados en los avances producidos a lo largo de los años en la construcción de métodos heurísticos y una serie de veinte instancias modeladas bajo el Problema de la Mochila, en este trabajo se presenta un desarrollo computacional para la resolución de un caso particular de este tipo de problemas, conocido como el Problema de la Mochila Multiopbjetivo, MO-KSP, para el cual se integran diferentes algoritmos constructivos, constructivos aleatorizados y de búsqueda

local, combinados con un Algoritmo Genético evolutivo, para producir así un Algorítmo Híbrido para la solución de este tipo de problemas.

En la siguiente sección de este trabajado se encontrará la revisión de literatura sobre temas relacionados. en la Sección 3 se realiza la descripción del problema a tratar y la formulación matemática que lo compone. Para la Sección 4, el algoritmo de solución es propuesto, seguido por la experimentación computacional realizada, presente en la Sección 5. Finalmente, la Sección 6 contiene las conclusiones y futuras implementaciones del estudio.

2. Revisión de literatura

Para los primeros planteamientos que se tiene registro sobre el Problema de la Mochila, se puede hablar del modelo clásico (KP) presente en [6], el cual ilustra un problema en el que se busca maximizar la utilidad que ciertos objetos escogidos representan, buscando así que estos cumplan un único valor de recurso máximo consumido. Posteriormente, este tipo de problema fue considerando muchas más restricciones, como es en el caso de [5], en el que se presenta un modelo de optimización para el Problema de la Mochila Multidimensional, también conocido como 0-1 KP o MKP. En este se presentan múltiples recursos, para los cuales cada objeto presenta un beso asignado, en donde la cantidad de restricciones, como la cantidad de combinaciones en la respuesta es mayor. Así mismo, estos desarrollos fueron llevados hasta el punto de obtener modelos multiobjetivo para el Problema de la Mochila, surgiendo así el MOKP presentado por [10], en donde se consideran los casos de maximización para diferente tipo de intereses.

A partir de los modelos que ha sido desarrollado a lo largo de los años, múltiples autores presentas diferente tipo de algoritmos para encontrar soluciones a problemas reales. Inicialmente, podemos encontrar en la literatura métodos exactos para la resolución del KP, así como el desarrollado por [9], en el cual se utiliza un método de ramificación Branch and Bound para explicar las posibles combinaciones de soluciones. Sin embargo, este desarrollo no resulta ser muy eficiente a la hora de trabajar con problemas grandes. Por esto, desde el siglo XX se vienen desarrollando algoritmos computacionales los cuales son de suma utilidad a la hora de solucionar problemas complejos, ya sea por su dimensión o por la cantidad de recursos requeridos. Estos desarrollos son conocidos como métodos heurísticos, procesos que permiten encontrar soluciones aproximadas de los problemas utilizando bajos recursos y tiempos de cómputo. estos han estado inspirados desde los comportamientos de los animales, hasta la misma evolución humana, buscando así diferentes formas de representación de los problemas reales. [8].

De acuerdo con la literatura, los algoritmos heurísticos están desarrollados en diferentes etapas. La primera de estas hace referencia la la construcción de soluciones iniciales, para lo cual se han desarrollado múltiples algoritmos constructivos. Este proceso se caracteriza por presentar resultados deterministas que potencian el comportamiento de los algoritmos más compleos, proporcionando un buen punto de partida en la búsqueda de soluciones óptimas. Luego, los metodos anteriores pueden ser potenciados mediante lo que se conoce como algoritmos aleatorizados, como podemos verlo en [12], en donde se realiza un proceso de selección aleatorio de mejores candidatos por medio de un método GRASP, el cual permite agregar un componente estocástico a los comportamientos iniciales de los métodos constructivos. Una tercera etapa de estos métodos se refiere a lo que se conoce como Búsqueda Local; este es un proceso por el cual una solución puede ser mejorada mediante diferente tipo de alteraciones de orden o elementos de la misma, buscando así escapar de óptimos locales. Una de los algoritmos más desarrollados de búsqueda local es la Busqueda de Vecindarios Varianble (VNS), como se presenta en [4], el cual se encarga de definir diferentes estructuras de vecindarios para cada solución presentada y así buscar mejorarla. Finalmente, los métodos evolutivos son otro de los desarrollos en el área de optimización por métodos heurísticos, el cual suele aprovechar los métodos anteriores para desarrollar algoritmos de gran eficiencia, especializados en problemas de optimización combinatoria, para encontrar estimar soluciones cerca de la región de optimalidad mediante procesos de selección de otras soluciones con mejores comportamientos.

Estos tipo de desarrollos anteriores ha sido ámpliamente aplicados al Problema de la Mochila. Por un lado, [3] presenta un algoritmo aleatorizado para la solución del MKP; a su vez, [11] y [7] presentan algoritmos basados en búsqueda local para el caso multidimencional y multiobjetivo. Por último, entre la gran cantidad de algoritmos propuestos, [1] ilustra e implementa un algoritmo genético aplicado al Problema de la Mochila Multidimensional, evidenciando en su trabajo el gran aporte de estos métodos en la búsqueda de soluciones óptimas.

3. Descripción del problema y formulación matemática

El Problema de la Mochila Multidimensional y multiobjetivo a resolver en este trabajo cuenta con un número F de funciones objetivo, en donde estas buscan ser maximizadas por la selección de diferentes objetivos, los cuales deben cumplir un número R de restricciones. Formalmente, el problema de la mochila multidimensional puede definirse de la siguiente manera:

Dado un conjunto $j = \{1, 2, \ldots, n\}$ de n items, se desea seleccionar un subconjunto de ítems con las siguientes características:

- Dado un conjunto de m tipos recursos de capacidad R, la cantidad de recursos utilizados de cada tipo K por cada ítem i, (a_{ik}), no puede exceder la capacidad dada b_k.
- Cada ítem puede ser incluido o no, pero no puede fraccionarse ni puede sobre ejecutarse.
- Se tiene un conjunto F de p funciones objetivo a optimizar donde w_{ih} es el peso del ítem i en la función objetivo h.

El problema puede ser formulado matemáticamente como:

$$\begin{array}{lllll} \text{máx} & \sum_{i=1}^{n} w_{ij} x_i & \forall & h \in F & \\ \text{s.t.} & & & & \\ & \sum_{j=1}^{n} a_{ik} x_i & \leq & b_k & \forall \quad k \in R \\ & x_i & \in & \{0,1\} & \forall i \in j \end{array} \tag{1}$$

4. Algoritmo de solución

El algoritmo de solución a implementar es un Algoritmo Genético Híbrido, el cual hace parte de los métodos evolutivos al tomar su inspiración en lo procesos de selección natural. Este a su vez es híbrido ya que integra elementos de Búsqueda Local en cada una de sus iteraciones para mejorar las soluciones. El algoritmo propuesto se presenta a continuación.

```
procedure Genetic_Algorithms()
    for i = 1 to population_size
        P[i] = generate_solution();
    end;
    for i = 1 to generations
        for j = 1 to number_of_children
            (x, y) ← selection(P[]);
            son[j] ← crossover(x, y);
            if random < prob_mutation do
                son[j] ← mutation();
            end;
        end;
        P[] ← update(P[]);
    end;
    return better solution in P[];
end Genetic_Algorithms
```

Figura 1: Algoritmo Genético, tomado del curso

En este algoritmo, método de soluciones iniciales fue desarrollados a partir de dos etapas. La primera de estas toma una porporción de soluciones iniciales brindadas por un algoritmos constuctivo aleatorizado, y las soluciones restantes para construr la población inicial son obtenidas mediante la aplicación de un vencidario de búsqueda VND, el cual intercambia en valor de dos soluciones, tomando como referencia una solución aleatoria de las ya generadas. En el método de selección se escogen dos individuos aleatoriamente de la población actual, buscando entre estos el

mejor, teniendo en cuenta si uno de estos domina al otro en las funciones objetivo. Este proceso se conoce como sorteo y es realizado dos veces para así tener dos padres a ser cruzados. En el proceso de cruce, generamos un valor aleatorio entre el número de items del problema, sea este t, y se toman los t primeros valores del padre uno y los restantes del padre 2, que se encuentran en una posición mayor a t. posteriormente se calcula un número aleatorio uniforme entre cero y uno y si este es menor que la probabilidad de mutación p, el hijo es mutado intercambiando dos de sus valores acutales en los items.

Este algoritmo, al ser un desarrollo híbido, implementa una búsqueda local MS-ILS, por sus cigles en inglés de Multi Start - Iterative Local Search, en la cual por medio de un facto de perturbación permite mejorar la solución actual. Esto genera una cantidad de soluciones no dominadas que son acumuladas hasta el final de la generación de los hijos. Cuando se tiene el número de iteraciones c requeridas, se realiza el proceso de actualización de las soluciones, tomando primero aquellas factibles no dominadas, luego aquellas factibles dominadas que pero que presentan mejores comportamientos que las otras, y así hasta completar el número de individios requeridos en la población. En el caso donde el número de soluciones factibles es menor a las requeridas, se completa la población con las soluciones infactibles que incumplen un número menor de restricciones.

El proceso anterior se realiza por un total de cinco minutos, en los cuales se generan tantas generaciones como sea posible, donde finalmente se aplica un proceso de factibilidad y dominancia para así devolver las soluciones no dominadas entre las que lograron sobrevivir al proceso evolutivo.

5. Experimentación computacional

5.1. Parametrización del algoritmo

El la construcción del algoritmo genético híbrido se cuenta con tres parámetros de suma importancia. Estos hacen referencia al número de individios en cada generación del algoritmo, n, el número de hijos a obtener en cada generación, c, y la probabilidad de mutación de un individuo,p. Por lo que de realizó una variación de dichos parámetros para ocho de las instancias del problema, siendo estas una muestra que representan los diferentes problemas en relación al número de restricciones. Así, se consideraron tres parámetros particulares para el análisis del comportamiento, como se muestra a continuación:

Variación parámetro n

En la Figura 2 podemos observar los resultados del algoritmo implementado para las ocho instancias, donde en la primera columna se pueden observar el número de restricciones del problema. Los criterios a evaluar en este análisis hacen referencia al número de soluciones no dominadas que se lograron generar y a la factibilidad de las soluciones.

Variación del número de individuos en cada generación (n)							
Restricciones	Iteración	p=0.2, n=100, c=100		p=0.2, n=150, c=100		p=0.2, n=200, c=100	
		Sol. No dominadas	Factibilidad	Sol. No dominadas	Factibilidad	Sol. No dominadas	Factibilidad
7	I3	2	Factible	3	Factible	6	Factible
	I4	3	Factible	6	Factible	5	Factible
15	I8	32	Factible	32	Factible	1	Factible
	I12	5	Factible	5	Factible	3	Factible
45	I13	16	Factible	18	Factible	18	Factible
	I14	5	Factible	15	Factible	12	Factible
10	I19	12	Factible	12	Factible	9	Factible
184	I20	10	Factible	4	Factible	8	Factible

Figura 2: Variación del parámetro n

Como se puede observar, al ralizar una varianción de $n =, 100, \quad , 150, \quad 200$, en las iteraciones consideradas se obtiene factibilidad en todos los casos. Además, Puede observarse que no siempre al considerar un número mayor de individuos en la población se obtienen un mayor número de soluciones no dominadas, pues se puede ver que para las dos primeras iteraciónes se tienen mejores resultados. Esto se debe a la limitación que se presenta en la variabilidad de las nuevas soluciones generadas.

Variación c

En la Figura 3 se presentan los resultados de la variación del número de hijos a generar en cada iteración, donde estos presentan una variación $c = 100, \quad , 150 \quad 200$, para obtener así una caracterización respecto al número de soluciones factibles y no dominadas

Variación del número de hijos en cada generación ©							
Restricciones	Iteración	p=0.2, n=100, c=100		p=0.2, n=200, c=150		p=0.2, n=200, c=200	
		Sol. No dominadas	Factibilidad	Sol. No dominadas	Factibilidad	Sol. No dominadas	Factibilidad
7	I3	2	Factible	1	Factible	4	Factible
	I4	3	Factible	2	Factible	4	Factible
15	I8	32	Factible	16	Factible	17	Factible
	I12	5	Factible	1	Factible	3	Factible
45	I13	16	Factible	1	Factible	7	Factible
	I14	5	Factible	14	Factible	16	Factible
10	I19	12	Factible	9	Factible	8	Factible
184	I20	10	Factible	7	Factible	8	Factible

Figura 3: Variación del parámetro c

Tal como en el caso anterior, el algoritmo genético híbrido presenta factibilidad en las ocho instancias consideradas, donde en relación con el número de soluciones no dominadas en cada iteración, se presentan mejores comportamientos para los valores más bajos del parámetro. Este comportamiento se presenta dado que al tener un número mayor de hijo, basados en el tiempo máximo estipulado para el problema, el número de generaciones que se construyen y analizan es menor, por lo que es probable que no se logre generar suficiente variabilidad para que el método escape de soluciones óptimas locales.

Variación parámetro p

Para la variacón de la probabilidad de mutación presente en el problema, en la Figura 4 pueden observarse los resultados para valores considerados de $p = 0{,}1, \quad , 0,2, \quad , 0{,}3$. Donde se analizan los aspectos mencionados anteriormente.

Variación de la probabilidad de mutación (p)							
Restricciones	Iteración	p=0.1, n=100, c=100		p=0.2, n=200, c=150		p=0.3, n=200, c=200	
		Sol. No dominadas	Factibilidad	Sol. No dominadas	Factibilidad	Sol. No dominadas	Factibilidad
7	I3	3	Factible	4	Factible	6	Factible
	I4	8	Factible	4	Factible	7	Factible
15	I8	12	Factible	17	Factible	12	Factible
	I12	8	Factible	3	Factible	6	Factible
45	I13	13	Factible	7	Factible	15	Factible
	I14	21	Factible	16	Factible	21	Factible
10	I19	11	Factible	8	Factible	13	Factible
184	I20	5	Factible	8	Factible	8	Factible

Figura 4: Variación del parámetro p

En esta tabla podemos ver que los comportamientos en la variación del parámetro de mutación no presenta un patrón de dominancia sobre el valor con mejores resultados. Esto se debe al hecho de que estamos trabajando bajo una estructura estocástica, en donde es necesario, para todos los casos, analizar comportamientos para diferentes corridas del modelo. Sin embargo, podemos observar que este parámetro presenta una influencia destacable en el número de soluciones no dominadas, lo cual demuestra que la consideración de este factor es de suma importancia a la hora de agregar diversificación en los resultados. Por último, como en los casos anteriores, se puede observar que la facibilidad de las soluciones se da en los ocho casos estudiados.

5.2. Evaluación de los comportamientos del algoritmo

Para analizar el comportamiento del algoritmo, se tuvieron en cuenta los parámetros $n = 100, \quad c = 100, \quad c = 0{,}2$, en donde se consideraron los tiempos de cómputo en relación con el número de soluciones factibles y no dominadas. Los resultados se presentan en la Figura 5.

Resultados Algoritmo Genético híbrido		
Iter	No dominadas	Tiempo
I1	14	300.01
I2	13	300.004
I3	2	300.432
I4	3	300.2
I5	5	300.002
I6	1	300.1
I7	23	300.124
I8	32	300.093
I9	12	300.501
I10	9	300.021
I11	1	300.123
I12	5	300.004
I13	16	302.032
I14	5	304.023
I15	16	300.023
I16	7	300.001
I17	3	300.231
I18	5	300.02
I19	12	300.154
I20	10	300.008

Soluciones Factibles	20

Figura 5: Resultados algoritmo genético híbrido implementado

En la tabla anterior podemos observar, dado que el tiempo máximo del algorimo es de cinco minutos, que los tiempos entre las iteraciones son muy similares, alcanzando estos el tope permitido. Esto muestra que el proceso se vio interrumplido por la limitante temporal propuesta, donde probablemente las iteraciones planeadas no lograron ejecutarse. Sin embargo, se puede observar que se presenta factibilidad en las veinte instancias consideradas, resultados favorables y esperados con este tipo de algoritmos, donde en el 45 % de los casos se obtuvo un número de soluciones no dominadas por encima de diez.

5.3. Comparaciones con otros métodos

5.4. Factibilidad y dominancia

Para el Problema de la mochila Multidimensional y MUiltiobjetivo estudiado en este trabajo se tuvo diferentes desarrollos de algoritmos previos. Estos hacen referencia a dos métodos constructivos, uno de estos enfocado en la satisfacción de soluciones, MC1, y en la potenciación de las utilidades, MC2. Adicionalmente, se desarrollaron dos algoritmos constructivos aleatorizados, los cuales estuvieron basados en el método del Ruidos y del algoritmo GRASP. Por otro lado, se consideraron dos algoritmos de búsqueda local, estos están basados en el algoritmo VND y MS-ILS.

En la figura 6 se observan los resultados para las veinte instancias del problema, en relación con el npumero de soluciones factibles encontradas por cada algoritmo, el número de soluciones no dominadas presentes en cada uno de ellos, y además, un diagrama ilustrativo donde se puede ver la dominancia de un método sobre otro en relación con el número de soluciones no dominadas que se generan.

	Número de soluciones no dominadas						
Iter	M.C.1	M.C.2	GRASP	Ruido	VND	MS-ILS	AG
I1	1	1	2	1	11	3	14
I2	1	1	1	2	5	2	13
I3	1	1	1	2	2	2	2
I4	1	1	2	1	2	6	3
I5	1	1	1	1	6	4	5
I6	1	1	2	1	2	6	1
I7	1	1	1	1	5	3	23
I8	1	1	1	2	5	5	32
I9	1	1	1	1	2	2	12
I10	1	1	1	1	1	3	9
I11	1	1	1	1	3	11	1
I12	1	1	1	1	1	3	5
I13	1	1	1	1	4	4	16
I14	1	1	1	1	1	1	5
I15	1	1	1	2	7	2	16
I16	1	1	1	2	1	2	7
I17	1	1	1	2	1	7	3
I18	1	1	2	2	8	7	5
I19	1	1	2	2	1	1	12
I20	1	1	1	1	1	1	10
Soluciones Factibles	17	4	9	18	18	18	20

Mayor frontera
AG
AG
No aplica
MS-ILS
VND
MS-ILS
AG
AG
AG
AG
MS-ILS
AG
AG
AG
AG
MS-ILS
VND
AG
AG

M.C.1	Método Constructivo con enfoque en restricciones
M.C.2	Método Constructivo con enfoque en utilidades
GRASP	Método Aleatorizado GRASP
Ruido	Método Aleatorizado Ruido
VND	Búsquela local VND con solución inicial constructiva
MS-ILS	Búsqueda local MS-ILS con Solución inicial constructiva
AG	Algoritmo genético Híbrido

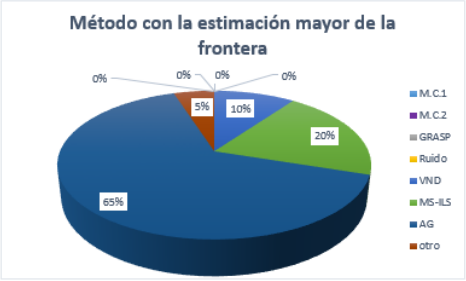

Figura 6: Factibilidad y soluciones no dominadas de los métodos implementados

Se puede observar que el algoritmos que presenta mayor factivilidad en las iteraciones es el Algoritmo Genético híbido desarrollado en este trabajo, pues este logra todas sus soluciones factibles. Además, es este algormitmo el que presenta un mayor número de soluciones no dominadas, siento este por iteración mejor que los otros algortmos en el 65 % de los casos. A su vez, se puede observar que los algoritmos aleatorizado del Ruido, el VND y el MS-ILS presentan una muy buena cantidad de iteraciones con soluciones factibles, donde los últimos dos presentan buenos resultados en la generación de soluciones no dominadas.

5.5. Tiempos de cómputo

En el análisis de algoritmos es muy importante considerar los tiempos de cómputo, ya que este puede ser un factor decisivo en la escogencia de un desarrollo sobre otro. Por esto, en la Figura 7 se presenta los tiempos de cómputo utilizados por cada uno de los algoritmos descritos anteriormente.

	Tiempos de corrida para los algoritmos implementados						
Iter	M.C.1	M.C.2	GRASP	Ruido	VND	MS-ILS	AG
I1	0.0191	0.0937	0.0285	0.0571	0.05785	0.078702	300.01
I2	0.0042	0.0128	0.0158	0.0277	0.3651	0.103025	300.004
I3	0.0044	0.0649	0.0272	0.1254	20.7229	0.481175	300.432
I4	0.0026	0.0155	0.036	0.1081	25.5686	0.6799	300.2
I5	0.0154	0.0381	0.096	0.5756	300.4772	10.7425	300.002
I6	0.003	0.0747	0.0793	0.6983	270.4792	29.163025	300.1
I7	0.0008	0.0029	0.0069	0.0267	0.5738	0.041225	300.124
I8	0.0006	0.0031	0.0085	0.037	0.3685	0.10595	300.093
I9	0.0012	0.0108	0.026	0.1367	10.6917	1.7696	300.501
I10	0.0022	0.0283	0.0273	0.1156	17.9355	6.21115	300.021
I11	0.0026	0.0285	0.1184	0.588	276.4005	55.532825	300.123
I12	0.0023	0.0287	0.1513	0.5522	163.6548	14.865225	300.004
I13	0.0028	0.0064	0.011	0.0252	300.0016	0.21775	302.032
I14	0.0009	0.0043	0.0094	0.0239	300.0049	0.133825	304.023
I15	0.0006	0.0167	0.0647	0.2533	3.6909	1.789675	300.023
I16	0.0012	0.016	0.0368	0.1264	1.7917	1.5941	300.001
I17	0.0022	0.093	0.1457	0.6027	61.7697	8.676525	300.231
I18	0.0128	0.0819	0.1361	0.7632	94.0763	45.857525	300.02
I19	0.002	0.0027	0.0303	0.0335	112.2676	0.2522	300.154
I20	0.0035	0.0028	0.031	0.1312	0.0774	0.048725	300.008

Figura 7: Tiempos de cómputo de los métodos implementados

Se puede ver una gran dominancia sobre los tiempos de cómputos para los algortimos constructivos y contructivos aleatorizados, esto dado que son desarrollos mucho más simples los cuales no requieren gran esfuerzo computacional. Estos resultados se deben, adicionalmente, al hecho de que los algoritmos de búsqueda Local y el Algoritmo genético híbriodo que han sido implmentados, funcionan con una solución inicial construida por alguno de los primeros métodos mencionados, lo que ocasiona que el tiempo de cómputo sea mayor. No obstante, es importante resaltar el buen

desempeño de estos algoritmos con tiempos mayores, pues por un lado tenemos un desarrollo en donde se poseee factibilidad en todos los resultados, como en otros en donde la aproximación de la frontera de Pareto es mejor al presentar mayor número de soluciones no dominadas.

5.6. Calidad en las soluciones

Para el análsisi en la calidad de las soluciones generadas por cada uno de los algoritmos anteriores, se tuvo en cuenta dos indicadores como criterios. El Indicador 1 se encarga de evauar la probabilidad de generar soluciones en la frontera de pareto en cada iteración que posee un método. Para el Indicador 2, loq ue se busca es estimar la probabilidad de generar soluciones en cada interación, donde estas sean no dominadas. Los resultados de estos indicadores se muestran en la Figura 8

	Indicadores de efectividad en la generación de soluciones													
	Indicador 1							Indicador 1						
Iteración	M.C.1	M.C.2	GRASP	Ruido	VND	MS-ILS	AG	M.C.1	M.C.2	GRASP	Ruido	VND	MS-ILS	AG
I1	0%	0%	0%	0%	50%	0%	50%	0%	0%	0%	0%	100%	0%	100%
I2	0%	0%	0%	0%	33%	33%	33%	0%	0%	0%	0%	100%	100%	100%
I3	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	100%	0%	0%
I4	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	100%	0%	0%
I5	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	100%	0%	0%
I6	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	100%	0%	0%
I7	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	100%
I8	0%	0%	0%	25%	25%	25%	25%	0%	0%	0%	100%	100%	100%	100%
I9	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	100%	0%	0%
I10	0%	0%	0%	0%	50%	0%	50%	0%	0%	0%	0%	100%	0%	100%
I11	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	100%	0%	0%
I12	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	100%	0%	0%
I13	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	100%
I14	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	100%
I15	0%	0%	0%	0%	0%	50%	50%	0%	0%	0%	0%	0%	100%	100%
I16	0%	0%	0%	0%	33%	33%	33%	0%	0%	0%	0%	100%	100%	100%
I17	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	100%	0%	0%
I18	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	100%	0%	0%
I19	0%	0%	0%	0%	50%	50%	50%	0%	0%	0%	0%	100%	100%	100%
I20	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	100%

Figura 8: Indicadores de calidad en la generación de soluciones

Se puede observar que tanto los métodos constructivos como los constructivos aleatorizados presentan comportamientos muy poco favorables, pues sus soluciones están dominadas por las de los dempas métodos, y además, estas no se encuentran en la frontera de pareto. Por esto, son los algoritmos de búsqueda local y el algoritmo genético desarrollados los que representan mejores soluciones. En el caso del indicador uno son los algortims VND y el Algoritmo genético los que poseen mayor número de soluciones en la frontera de Pareto; este efectivo comportamiento de los dos métodos anteriormente destacados se presenta de igual manera para el indicador 2, pues son estos mismos algoritmos los que poseen un mayor número de soluciones no dominadas para cada una de las iteraciones.

6. Conclusiones

En este trabajo se propone un Algoritmo Genético híbrido para el Problema de la Mochila Multidimensional y Multiobjetivo. Por medio de los resultados encontrados, y en comparación con algoritmos desarrollados previamente, se puede concluir que los Algoritmos genéticos, en especial aquellos que son híbridos, presentan un muy buen comportamiento en relación a la generaciones de soluciones factibles, no dominadas, donde además se resalta una buena aproximación a la frontera de Pareto. Sin embargo, es de resaltar que este tipo de algoritmos presentan un tiempo de cómputo superior a otro tipo de desarrollos más sencillos, por lo es que es importante, basados en el objetivo a la hora de aplicar un método heurístico, definir qué tipo de resultados se necesitan obtener. Adicionalmente, podemos ver el gran aporte que genera el utilizar métodos de busca local, pues estos ayudan en gran medida a eliminar la dependecia a las condiciones iniciales que tienen en los métodos constructivos y trabajar de manera más focalizada que en un método constructivo aleatorizado.

Dado que en este trabajo se realizó bajo procesos de búsqueda y construcción de soluciones iniciales por medio de métodos aleatorios, se recomienda para próximos trabajos considerar nuevos vecindarios o perturbaciones donde se puedan evaluar mayor número de soluciones en un tiempo

menor, para así potenciar los algoritmos de búsqueda local. A su vez, buscando mejorar los tiempos de cómputo del Algoritmo genético, se espera definir mejores criterios de parada para la obtención de nuevas generaciones, consideranto en esto los valores de desviación entre una nueva solución obetenida y la presente en la generación anterior, como también en el diseño de algoritmos más efectivos desde su estructura de desarrollo interno, diminuyendo su complejidad.

Referencias

[1] Chu, P. C., and Beasley, J. E. A genetic algorithm for the multidimensional knapsack problem. *Journal of heuristics 4*, 1 (1998), 63–86.

[2] Cunquero, R. M. Procedimentos metaheurísticos en optimización combinatoria.

[3] de Almeida Dantas, B., and Cáceres, E. N. Sequential and parallel implementation of grasp for the 0-1 multidimensional knapsack problem. *Procedia Computer Science 51* (2015), 2739–2743.

[4] Duarte, A., Pantrigo, J. J., Pardo, E. G., and Mladenovic, N. Multi-objective variable neighborhood search: an application to combinatorial optimization problems. *Journal of Global Optimization 63*, 3 (2015), 515–536.

[5] Fréville, A. The multidimensional 0–1 knapsack problem: An overview. *European Journal of Operational Research 155*, 1 (2004), 1–21.

[6] Fuentes-Penna, A., Vélez-Díaz, D., Moreno-Gutiérrez, S., Martínez-Cervantes, M., Sánchez-Muñoz, O., et al. Problema de la mochila (knapsack problem). *XIKUA Boletín Científico de la Escuela Superior de Tlahuelilpan 3*, 6 (2015).

[7] Hanafi, S., Lazić, J., Mladenović, N., and Wilbaut, C. Variable neighbourhood decomposition search with bounding for multidimensional knapsack problem. *IFAC Proceedings Volumes 42*, 4 (2009), 2018–2022.

[8] Kokash, N. An introduction to heuristic algorithms. *Department of Informatics and Telecommunications* (2005), 1–8.

[9] Kolesar, P. J. *A Branch and Bound Algorithm for the Knapsack Problem.* Management Science 13(9):723-735, 1967. http://dx.doi.org/10.1287/mnsc.13.9.723.

[10] Lust, T., and Teghem, J. The multiobjective multidimensional knapsack problem: a survey and a new approach. *International Transactions in Operational Research 19*, 4 (2012), 495–520.

[11] Paz Duarte, P. D., and Cepeda, D. F. Algoritmo metaheurístico basado en el enfoque mos para resolver el problema 0-1 de la mochila cuadrática en instancias de alta dimensionalidad.

[12] Reynolds, A. P., and De la Iglesia, B. A multi-objective grasp for partial classification. *Soft Computing 13*, 3 (2009), 227–243.