GRASP

Author 1: Juan David Castaño Vargas, Author 2: Juan Camilo Galindo
Magister of Engineer of Systems & Computing, Universidad Tecnológica de Pereira, Pereira, Colombia
E-mail: juandavid.castano@utp.edu.co, j.galindo@utp.edu.co

***Abstract*— GRASP is a heuristic approach to solving combinatorial optimization problems. The acronym stands for "Greedy Randomized Adaptive Search Procedure," which means it combines a greedy algorithm with an adaptive search process to find an optimal or near-optimal solution in a reasonable amount of time.**

The GRASP algorithm starts by selecting an initial solution at random. Then, a local search process is applied to improve the solution. After several iterations, the best solution found is chosen. At each iteration, the algorithm selects a partial solution using greedy criteria that balances solution quality and randomness. This is done to avoid getting stuck in local optima.

Once a partial solution is selected, a local search procedure is applied that attempts to improve the solution by removing or exchanging components of the current solution. This process is repeated until a better solution is found or a time limit or iterations is reached.

***Key Word* — Heuristic algorithm, Combinatorial optimization, Random initial solution, Local search process, Greedy criterion, Global optimal solution Experimentation, Artificial Intelligence.**

I. INTRODUCTION

GRASP, which stands for "Greedy Random Adaptive Search Procedure", is a heuristic approach to solving combinatorial optimization problems. It is a metaheuristic method that combines a greedy algorithm with an adaptive search process to find an optimal or near-optimal solution in a reasonable amount of time.

Combinatorial optimization problems are those in which the objective is to find the best solution among a finite set of possible solutions. These issues arise in a wide variety of fields, including logistics, scheduling, resource allocation, and many others. However, finding an optimal solution to these problems is often very difficult, and in many cases simply not feasible due to the large number of possible solutions.

GRASP addresses this challenge using a two-phase approach to find an optimal or near-optimal solution. In the first phase, a random initial solution is generated. In the second phase, a local search process is applied to the initial solution to improve its quality. This process is repeated for a certain number of iterations or until a stopping criterion is met.

The local search process in GRASP is designed to explore the space of feasible solutions in a systematic way. It does this by iteratively modifying the current solution to improve its quality. Modifications are made by removing or swapping solution components. The goal is to find a local optimum that is better than the current solution.

One of GRASP's strengths is its ability to balance exploration and exploitation. The algorithm combines greedy criteria with a random process to ensure that it explores a wide range of solutions while also exploiting promising areas of the search space. This helps prevent the algorithm from getting stuck in local optima and increases the probability of finding a global optimum.

In this document we will use this algorithm to study its behaviour in a specific problem called "Knapsack Problem", likewise we will also tell our process, experience and we will give our conclusions carrying out the aforementioned.

II. DEPLOYMENT

To deploy the GRASP algorithm, we follow these steps:

A. Initialise the problem: We define the knapsack problem and initialise the decision variables, for this we use 3 schemes with the same sensitivity indicator which is given by the formula (C*0.5) +((15-V) *0.5), where C is cost and V the volume of each possible item inside the knapsack.

B. Define the objective function: we define the objective function to maximise the total value of the items in the knapsack subject to the capacity constraint.

C. Set the parameters for the GRASP algorithm: We set the parameters for the GRASP algorithm, such as the number of iterations, the size of the candidate list and the threshold for selecting items.

D. Generate initial solutions: We generate initial solutions using the random greedy algorithm. For each solution, use a random threshold to select the items to be included in the knapsack.

E. Run the GRASP algorithm: We run the GRASP algorithm by iteratively generating solutions, selecting the best solution from the candidate list, and updating the threshold.

F. Implement path linking: We implement path linking to improve the quality of the solutions. Path linking involves creating a path between two solutions by iteratively selecting an item to add to or remove from the rucksack. This process continues until the two solutions become identical.

G. Repeat steps 4 to 6 until convergence: We repeat steps 4 to 6 until the solutions converge to a near-optimal solution.

H. Select the best solution: We select the best solution obtained from the GRASP algorithm with path reconnection as the final solution.

I. Evaluate the solution: We evaluate the final solution to ensure that it satisfies the capacity constraint of the rucksack.

Following these steps, we obtain a few results which we will study in the next chapter.

III. REVIEW

As we have been explaining before, this algorithm is an efficient and powerful technique to solve the knapsack problem and other combinatorial optimization problems. This combination of techniques allows the algorithm to efficiently explore the search space and converge towards high-quality solutions. However, the performance of the GRASP algorithm with path relinking depends to a large extent on the quality of the solutions generated by the heuristics and the path relinking method. Furthermore, the tuning of the parameters is an important factor that can significantly affect the performance of the algorithm.

Next, we are going to expose our results obtained during the practice, accompanied by some graphic and statistical resources to see in a better way how the algorithm behaves throughout the study.

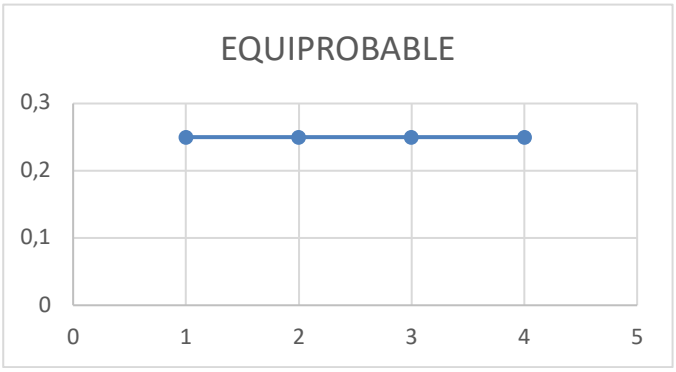

Figure 1: Results of equally probable

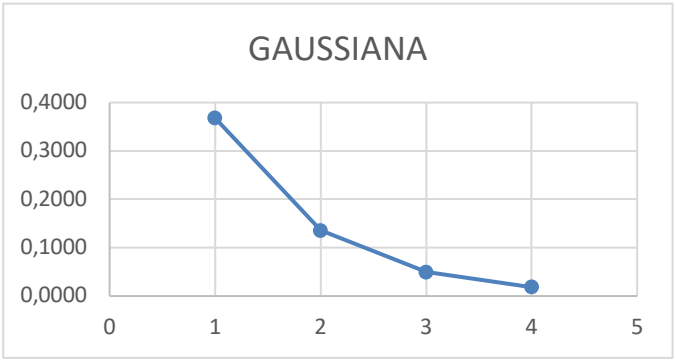

Figure 2: Results of Gauss method

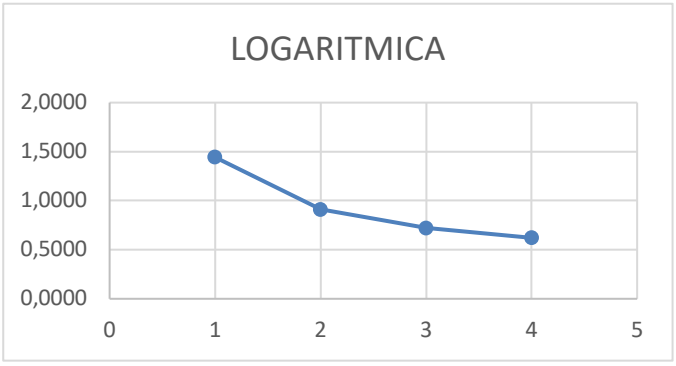

Figure 3: Results of logarithm method

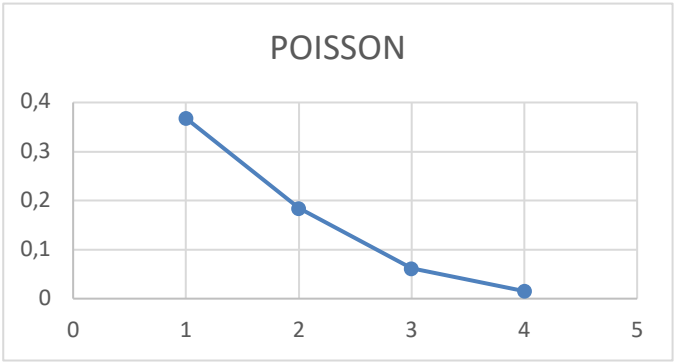

Figure 4: Results of Poisson method

EQUIPROBABLE					GAUSSIANA				
1	0,25				1	0,3679			
2	0,25				2	0,1353			
3	0,25				3	0,0498			
4	0,25				4	0,0183			

LOGARITMICA				POISSON			
1	1,4427			1	0,37		
2	0,9102			2	0,18		
3	0,7213			3	0,06		
4	0,6213			4	0,02		

Table 1: Results of methods

With the results obtained from the previous graphs, we can say that:

Equiprobable distribution: In this scheme, all objects have the same probability of being selected in each iteration of the construction of initial solutions. This means that no preference is given to any object. The equiprobable distribution is easy to implement and can work well in some cases, but it can lead to suboptimal solutions if some objects are more important than others.

Gaussian distribution: In this scheme, the probability of selecting an object is determined by a normal (Gaussian) distribution centred on the mean value of the available objects and with a standard deviation determined by a parameter. Objects closer to the mean value are more likely to be selected. The Gaussian distribution can generate initial solutions that are closer to the optimal value but can be more difficult to fit due to the need to fit the appropriate standard deviation.

Log distribution: In this scheme, the probability of selecting an object is determined by the logarithm of the value of the object. This means that higher value items have a higher probability of being selected than lower value items. The logarithmic distribution can work well for knapsack problems with objects that are very different in value but can lead to suboptimal solutions if there are many objects with similar values.

Poisson Distribution: In this scheme, the probability of selecting an object is determined by a Poisson distribution, which measures the probability of an event occurring in a given period of time. In the context of constructing random initial solutions in GRASP, the Poisson distribution is used to determine the number of objects that are selected in each iteration. The Poisson distribution can be useful in knapsack problems where the number of objects is very large, but it can be difficult to fit because of the need to determine the proper distribution parameter.

In summary, we can review that the equiprobable distribution is the only one that presents a constant probability line, while the remaining distributions have a somewhat similar probability line.
Next, we will review the evolution of the 3 schemes used in this practice.

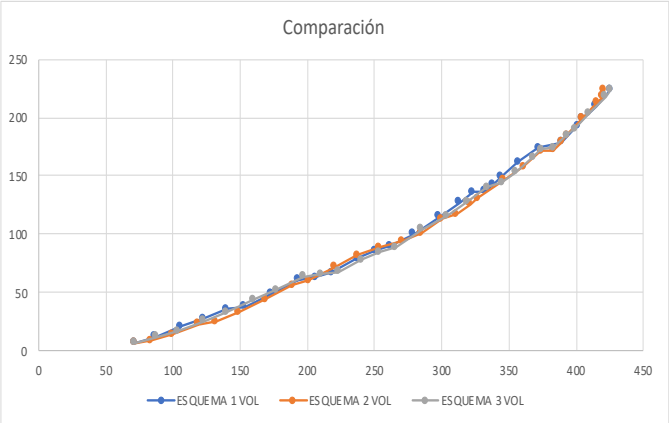

Figure 5: Comparation of schemes

ESQUEMA 1		ESQUEMA 2		ESQUEMA 3	
FO	VOL	FO	VOL	FO	VOL
71	7	71	7	71	7
87	12	84	9	88	12
106	21	100	14	104	17
123	28	119	23	123	26
140	36	132	25	140	33
153	38	149	33	160	44
173	49	169	44	177	52
193	61	189	56	197	64
206	63	201	60	210	66
218	67	220	72	223	68
237	79	237	82	240	78
251	86	254	89	254	85
262	90	271	94	266	89
279	100	285	101	285	104
298	115	300	113	304	116
313	127	311	117	319	128
323	136	321	126	334	140
332	137	327	131	345	144
338	142	346	146	355	153
344	149	361	158	368	166
357	162	374	171	374	173
372	174	383	172	383	174
389	179	389	179	393	185
401	193	393	185	399	190
405	199	404	199	410	204
415	210	416	213	422	218
426	224	420	218	426	224
		421	224		

Table 2: Comparation of schemes

IV. CONCLUSIONS

In conclusion, the GRASP algorithm with path relinking is a powerful heuristic method for solving the knapsack problem. It is a metaheuristic algorithm that combines the strengths of greedy randomized algorithm, local search, and path relinking to find near-optimal solutions.

The GRASP algorithm uses a candidate list of items and a random threshold to generate initial solutions, which are then improved by local search. Path relinking is used to further improve the quality of the solutions by creating a path between two solutions and iteratively selecting items to add or remove from the knapsack.

The performance of the GRASP algorithm with path relinking depends on several factors such as the size of the problem, the parameters of the algorithm, and the quality of the initial solutions. However, empirical studies have shown that the GRASP algorithm with path relinking can provide high-quality solutions for large-scale knapsack problems within reasonable time.

Overall, the GRASP algorithm with path relinking is a useful tool for solving the knapsack problem and can be used in various applications such as resource allocation, production planning, and software development.

REFERENCES

Scientific publication references:

[1]. *Zhang, Q., Chen, H., & Yan, S. (2017). An improved GRASP algorithm with path relinking for the 0-1 multidimensional knapsack problem. In 2017 International Conference on Logistics, Informatics and Service Science (LISS) (pp. 75-80). IEEE.*

[2]. *Hu, Y., Wang, J., Li, S., Zhang, L., & Li, C. (2016). A novel hybrid algorithm based on GRASP and path relinking for knapsack problems. In 2016 3rd International Conference on Information Science and Control Engineering (ICISCE) (pp. 312-316). IEEE.*

Book References:

[3]. Laguna, M., & Martí, R. (2018). Iterated local search and GRASP: A unified reactive variable neighbourhood search framework. Springer.

[4]. *VoB, S., Martello, S., Osman, I. H., & Roucairol, C. (Eds.). (2010). Metaheuristics: Progress in complex systems optimization (pp. 277-299). Springer.*

[5]. *Resende, M. G. C., & Ribeiro, C. C. (2005). GRASP with path-relinking: recent advances and applications. In Handbook of metaheuristics (pp. 251-284). Springer, Boston, MA.*

Rules:

[6]. *IEEE Guide for Application of Power Apparatus Bushings*, IEEE Standard C57.19.100-1995, Aug. 1995.