International **J**ournal of
Soft **C**omputing and **S**oftware **E**ngineering (JSCSE)

Vol.4,No.5, 2014
Published online: May 25, 2014

e-ISSN: 2251-7545
DOI: 10.7321/jscse.v4.n5.1

An Estimation of Distribution Algorithm for solving the Knapsack problem

[1]Ricardo Pérez, [2]S. Jöns, [3]Arturo Hernández, [4]Carlos A. Ochoa
*[*1,] PICYT-CIATEC A.C., León City, México*
[2,] CONACYT, México City, México
[3]CIMAT A.C., Guanajuato City, México
[4]UACJ, Ciudad Juárez City, México
Email: [1]rperez.picyt@ciatec.mx, [2]jons_sanchez@hotmail.com, [3]artha@cimat.mx, [4]alberto.ochoa@uacj.mx

Abstract. **The knapsack problem, a NP-hard problem, has been solved by different ways during many years. However, its combinatorial nature is still interesting for many academics. In this paper, an Estimation of Distribution Algorithm is applied for solving the Knapsack problem. KEDA for simplicity is called. It contains a probabilistic model of type chain for sampling new offsprings to solve the problem. In addition, we use a Greedy Algorithm and a Genetic Algorithm to compare the performance of the KEDA algorithm. According to the experiments, the genetic algorithm and the KEDA provide good solutions, but the performance from this evolutionary algorithm was able to give better results.**

Keywords*: Estimation of Distribution Algorithm, Knapsack problem, genetic algorithm, greedy algorithm.*

* Corresponding Author:
Ricardo Pérez Rodríguez,
Posgrado Interinstitucional en Ciencia y Tecnología PICYT,
CIATEC, A.C., México,
Email: rperez.picyt@ciatec.mx Tel:+52-462-4906221

1. Introduction

Evolutionary and meta-heuristic algorithms have been extensively used as search and optimization tools during this decade in several domains from science to engineering, and others. Many demanding applications that involve the solution of optimization problems of high complexity, a lot of these belonging to a special class of problems called NP-hard have been solved by various methods [1]. Evolutionary and meta-heuristic algorithms are now considered among the best tools to find good solutions with a reasonable investment of resources.

Estimation of Distribution Algorithms (EDAs), introduced by Mühlenbein and Paaβ [2] have been used satisfactorily to solve complex combinatorial optimization problems. Chen et al [3], Liu et al [4] and Pan and Ruiz [5] can be consulted.

Disadvantages of EDAs such as loss of diversity and insufficient use of location information of solutions have been tackled successfully by incorporating other methods such as Genetic Algorithms (GAs) during the evolutionary process. Chen et al [6] use this approach.

Several works have been done in order to capture the problem structure with more precision. Advanced probabilistic models to solve combinatorial problems through EDAs have been proposed attempting to integrate higher order interactions to enhance the solution quality. Wang et al [7] and Chen et al [8] have contributed on it.

The major procedure of an EDA is listed as follows.

International **J**ournal of
Soft **C**omputing and **S**oftware **E**ngineering (JSCSE)

Vol.4,No.5, 2014
Published online: May 25, 2014

e-ISSN: 2251-7545
DOI: 10.7321/jscse.v4.n5.1

Step 1. Set the generation index g = 0. Initialize an initial population S(0) of size M.

Step 2. Select a subset D from S(g) of size N, where N≤M.

Step 3. Establish a probabilistic model P which somehow describes the distribution characteristics of D.

Step 4. Generate a set K of new individuals by sampling P.

Step 5. Select the best individuals from K∪ S(g) and assign them to the next generation S(g+1).

Step 6. Let g = g+1. If g<GN, where GN is the maximum number of generations return step 2. Otherwise, output the best solution in S(g).

The Knapsack problem is a classical combinatorial problem [9][10]. It can be described as follows: "Imagine taking a trip to which you can only carry a backpack that, logically, has a limited capacity. Given a set of items, each with a weight and a value, determine the number of each item to include in a bag so that the total weight is less than a given limit and the total value is as large as possible", this problem can be considerate as NP-easy problem but some studies show that the Knapsack problem is an NP-hard problem [11].

In the present paper we introduce the Estimation of Distribution Algorithm for solving the Knapsack problem called KEDA for simplicity. The experiments were made on four types of instances from uncorrelated to subset-sum. All these instances probe the algorithm varying the parameters of the profits and the weight. It was necessary to have a comparison point for the KEDA and was used the Greedy Algorithm [9], this is a deterministic algorithm who gives an approximate result for the Knapsack problem. In addition, a Genetic Algorithm or GA was used in order to compare the performance among them.

2. The Knapsack problem

The Knapsack problem [10] is the typical combinatorial problem that has been studied since many years ago and was proved that it is a NP-hard problem [12]. The basic problem is the 0-1 Knapsack problem or Binary Knapsack problem and it has a search space of $2^n - 1$ possible solutions.

The Knapsack Problem can be described as follows: "there are *n* objects, each of this objects have a profit and weight, and needs to select those whose sum of their benefits is maximized subject to the sum of the weight of the same objects should not exceed an amount determined". It can be formulated mathematically by numbering each of its objects or items from 1 to *n* and introducing it to a vector of binary variables = 1,2,3,...,*n*, where each variable represented here will take the value 1 or 0 depending on whether it is selected or not. The solution to the Knapsack problem is select a subset of objects from the binary vector, solution vector, that satisfies the constraint on the equation (2) and the same time maximize the objective function on the equation (1).

$$z = \sum_{j=1}^{n} p_j x_j \tag{1}$$

$$\sum_{j=1}^{n} w_j x_j \leq c \qquad x_j = \begin{cases} \text{if the } j \text{ object is selected} \\ 0 \text{ otherwise} \end{cases} \tag{2}$$

where
z represents the profit
j represents the j-th object
x_j indicates whether the j object is part of the solution
p_j is the j-th object profit
w_j is the j-th object weight

International **J**ournal of
Soft **C**omputing and **S**oftware **E**ngineering (JSCSE)

Vol.4,No.5, 2014
Published online: May 25, 2014

e-ISSN: 2251-7545
DOI: 10.7321/jscse.v4.n5.1

c is the volume or capacity of the knapsack

3. Types of Knapsack problem instances

The Knapsack Problem is affected by the relationship between the profit and the weight of the objects; these types of instances are the following:

Uncorrelated: the profits and weight are distributed uniformly between one and a maximum T number.

$$p_j \in [1,T];\ w_j \in [1,T] \tag{3}$$

Weakly correlated: the weight is distributed uniformly between one and a maximum T number and the profits are distributed uniformly around the weight and an R ratio.

$$w_j \in [1,T];\ p_j \in [w_j - R, w_j + R] \tag{4}$$

Strongly correlated: the weight is uniformly distributed between one and a maximum T number; the profits are the weight plus one K constant.

$$w_j \in [1,T];\ p_j = w_j + K \tag{5}$$

Subset-sum: the profits and weight have the same value and are distributed uniformly between one and a maximum T number.

$$w_j \in [1,T];\ p_j = w_j \tag{6}$$

4. The Greedy Algorithm

This algorithm gives an intuitive approach considering the profit and weight of each item; it is known as the efficiency which is based on the Equation (7). The objective is to try to put the items with highest efficiency into the Knapsack. It is necessary sort all the items based on the efficiency, using the Equation (8), before to apply the Greedy algorithm to the problem.

$$e_j = \frac{p_j}{w_j} \tag{7}$$

$$\frac{p_0}{w_0} \geq \frac{p_1}{w_1} \geq \cdots \geq \frac{p_n}{w_n} \tag{8}$$

5. KEDA for the knapsack problem

Our approach is to use the MIMIC algorithm to build the probabilistic graph model. Introduced by De Bonet et al [13], the MIMIC algorithm uses a chain structured probabilistic model where the probability distribution of all the variables except the head node is conditioned on the value of the variable preceding them in the chain. It means a marginal univariate function and *n-1* pairs of conditional density functions to build the probabilistic graph model.

Solution representation: any solution of the problem mentioned should be a specific binary vector that represents the objects in the knapsack. Thus, a solution can be expressed by 0 or 1 for each object according to section 2.

Probability model: In this paper, the probability model is designed as a probability matrix. The element p_j of the probability matrix represents the probability that the object *j* be loaded in the knapsack. For all *j* (*j = 1, 2, ... n*), p_j is initialized as

International **J**ournal of
Soft **C**omputing and **S**oftware **E**ngineering (JSCSE)

Vol.4,No.5, 2014
Published online: May 25, 2014

e-ISSN: 2251-7545
DOI: 10.7321/jscse.v4.n5.1

$$p_j(0) = \frac{\sum_{j=1}^{n} x_j = 1}{n} \quad (9)$$

where *n* represents the number of elements. Via sampling according to the probability matrix new promising individuals may be generated.

6. Experiments

To test the KEDA was used the Generator of Knapsack Test Instances [14]; it requires the number of elements and the coefficients range to generate a test instance. We generate the four types of test instances described, and was used the same parameters for each. Each algorithm was run 100 times for obtaining their average and standard deviation.

The Table 1 shows the parameters used in this research.

Table 1. Parameters used for solving the Knapsack problem

Parameters	Values
Generator of Knapsack Test Instances	
Number of items or elements	50
Range of Coefficients	1000
Number of instances	100
Number of test in series	1000
Genetic Algorithm	
Type of selection	Tournament of size 2
Cross rate	80%
Mutation rate	10%
KEDA	
Probabilistic Model	Structure of type chain

7. Results

We show the results obtained by testing each type instances with the different algorithms, i.e., the average, the best and the worst profit, and their fitness's standard deviation for each algorithm. We also show the algorithms behavior through some graphics.

Uncorrelated: the Table 2 depicts the results for all algorithms where the KEDA offers the best std. deviation and the best average.

Table 2. Results for uncorrelated instance

Algorithm	Worst	Best	Average	Std. deviation
Greedy Algorithm	1774	1774	1774	-
Genetic Algorithm	0	2067	1668	228
KEDA	0	2067	2009	103

- same result in any interaction, there is no exist std. deviation

The Figure 1 depicts the performance for each algorithm in each interaction.

International **J**ournal of
Soft **C**omputing and **S**oftware **E**ngineering (JSCSE)

Vol.4,No.5, 2014
Published online: May 25, 2014

e-ISSN: 2251-7545
DOI: 10.7321/jscse.v4.n5.1

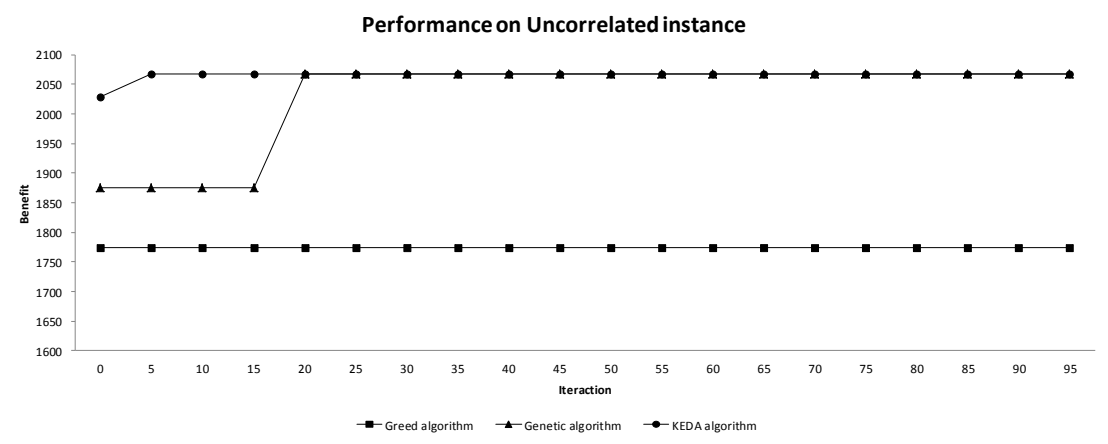

Figure 1. Trials for uncorrelated instance

Weakly correlated: the Table 3 details the results for all algorithms where the KEDA again offers the best std. deviation and the best average.

Table 3. Results for weakly correlated instance

Algorithm	Worst	Best	Average	Std. deviation
Greedy Algorithm	658	658	658	-
Genetic Algorithm	0	935	811	90
KEDA	0	935	890	55

- same result in any interaction, there is no exist std. deviation

The Figure 2 details the performance for each algorithm in each interaction.

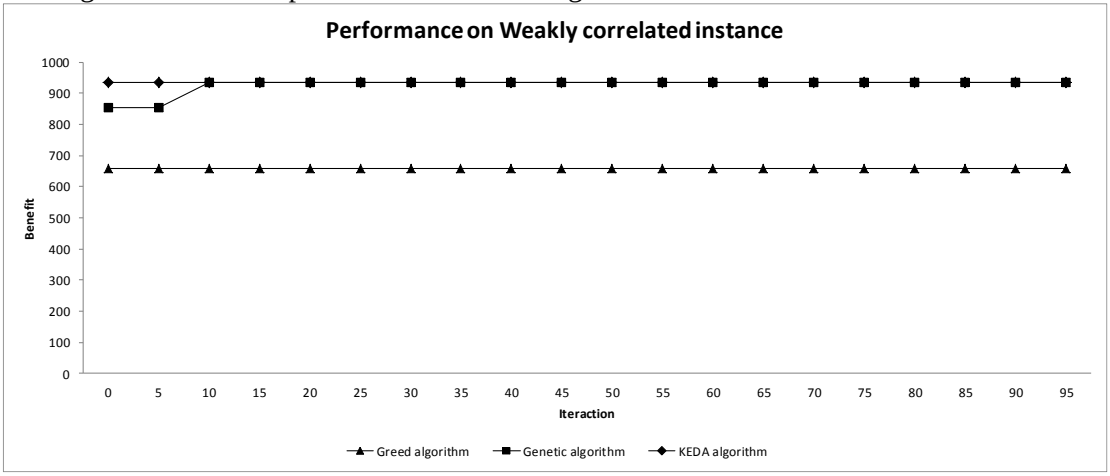

Figure 2. Trials for weakly correlated instance

Strongly correlated: the Table 4 shows the results for all algorithms where the KEDA again offers the best std. deviation and the best average.

Table 4. Results for strongly correlated instance

Algorithm	Worst	Best	Average	Std. deviation

International **J**ournal of
Soft **C**omputing and **S**oftware **E**ngineering (JSCSE)

Vol.4,No.5, 2014
Published online: May 25, 2014

e-ISSN: 2251-7545
DOI: 10.7321/jscse.v4.n5.1

Greedy Algorithm	828	828	828	-
Genetic Algorithm	0	1034	941	109
KEDA	0	1034	1019	15

- same result in any interaction, there is no exist std. deviation

The Figure 3 shows the performance for each algorithm in each interaction.

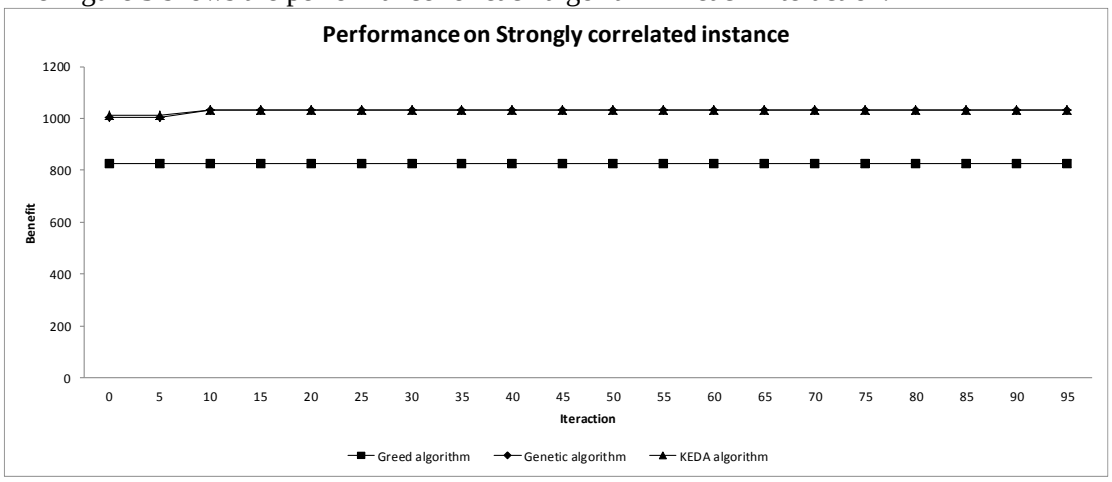

Figure 3. Trials for strongly correlated instance

Subset-sum: the Table 5 presents the results for all algorithms where the KEDA again offers the best std. deviation and the best average.

Table 5. Results for subset-sum instance

Algorithm	Worst	Best	Average	Std. deviation
Greedy Algorithm	573	573	573	-
Genetic Algorithm	0	996	914	107
KEDA	0	996	988	11

- same result in any interaction, there is no exist std. deviation

The Figure 4 presents the performance for each algorithm in each interaction.

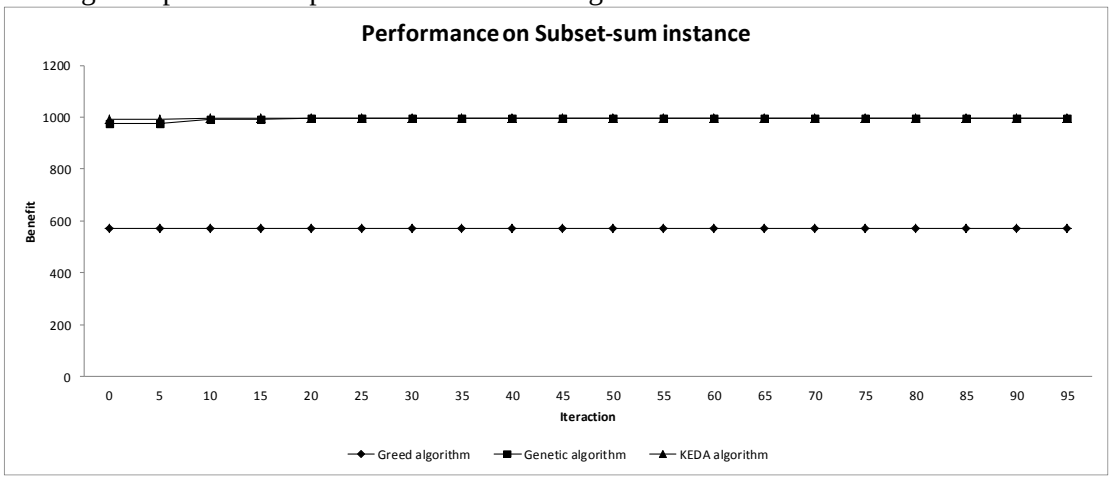

Figure 4. Trials for subset-sum instance

International **J**ournal of
Soft **C**omputing and **S**oftware **E**ngineering (JSCSE)

Vol.4,No.5, 2014
Published online: May 25, 2014

e-ISSN: 2251-7545
DOI: 10.7321/jscse.v4.n5.1

8. Conclusions

There are many evolutionary algorithms and meta-heuristics to solve the Knapsack problem; in this work we introduce the KEDA algorithm which is an evolutionary algorithm. The experiments were designed with the same parameters for the three algorithms to give them the same characteristics in order to be equal between them.

We can see in all the graphics the algorithms behavior, and we can observe that the Greedy Algorithm was the worst because it always yields the same result for any interaction. The KEDA and the Genetic Algorithm yield more consistent results, its graphs show that these algorithms in the most of the cases give good results.

So we can conclude that the KEDA is an alternative to solve the Knapsack problem, because each time that it's run, it gives a good solution, and this solution always is better that the solutions obtained by the other algorithms. Overall the results present a low standard deviation.

References

[1] McDuff-Spears W, Using neural networks and genetic algorithms as heuristics for NP-complete problems, Thesis of Master of Science in Computer Science, George Mason University, Virginia, USA, 1989.

[2] Mühlenbein H, Paaß G, "From recombination of genes to the estimation of distributions: I. binary parameters", in Parallel Problem Solving from Nature PPSN IV, Voigt H, Ebeling W, Rechenberg I, Schwefel H, Eds., Berlin: Springer, pp. 178–187, 1996.

[3] Chen S, Chen M, Chang P, Zhang Q, Chen Y, "Guidelines for developing effective Estimation of Distribution Algorithms in solving single machine scheduling problems", Expert Systems with Applications, vol. 37, pp. 6441-6451, 2010.

[4] Liu H, Gao L, Pan Q, "A hybrid particle swarm optimization with estimation of distribution algorithm for solving permutation flowshop scheduling problem", Experts Systems with Applications, vol. 38, pp. 4348-4360, 2011.

[5] Pan Q, Ruiz R, "An estimation of distribution algorithm for lot-streaming flow shop problems with setup times", Omega, vol. 40, pp. 166-180, 2012.

[6] Chen S, Chang P, Cheng T, Zhang Q, "A Self-guided Genetic Algorithm for permutation flowshop scheduling problems", Computers and Operations Research, vol. 39, pp. 1450-1457, 2012.

[7] Wang L, Wang S, Xu Y, Zhou G, Liu M, "A bi-population based estimation of distribution algorithm for the flexible job-shop scheduling problem", Computers and Industrial Engineering, vol. 62, pp. 917-926, 2012.

[8] Chen Y, Chen M, Chang P, Chen S, "Extended artificial chromosomes genetic algorithm for permutation flowshop scheduling problems", Computers and Industrial Engineering, vol. 62, pp. 536-545, 2012.

[9] Kellerer H, Pferschy U, Pisinger D, Knapsack Problems, Springer, Berlin, Germany, 2004.

[10] Silvano M, Toth P, Knapsack Problem, Algorithms, and Computer Implementations, John Wiley and Sons, New York, USA, 1990.

[11] Garey M, David S, Computers and Intractibility: A Guide to the Theory of NP-Completeness I, 1979.

[12] Pisinger D, "Where Are The Hard Knapsack problems?", Computers and Operations Research, vol. 32, pp. 2271-2282, 2005.

[13] De Bonet J, Isbell C, Viola P, "MIMIC: Finding Optima by Estimation Probability Densities", Advances in Neural Information Processing Systems, vol. 9, 1997.

[14] Pisinger D, "Core Problems in Knapsack Algorithms", Operations Research, vol. 32, pp. 2271-2282, 2005.