Chapter 9

Solving the 0–1 Knapsack Problem with EDAs

R. Sagarna P. Larrañaga
Department of Computer Science and Artificial Intelligence
University of the Basque Country
{ccbsaalr, ccplamup}@si.ehu.es

Abstract In this chapter we present several approaches to the 0-1 knapsack problem based on Estimation of Distribution Algorithms. These approaches use two different types of representation, three methods for obtaining the initial population and two different methods for handling the problem's constraints. Experimental results for problems of different sizes are given.

Keywords: knapsack problem, Estimation of Distribution Algorithms, binary representation, permutation representation

1. Introduction

The knapsack problem can be described as selecting from among various items that could be placed in a knapsack those items which are most useful given that the knapsack has limited capacity. Knapsack problems have been intensively studied because of their simple structure and because they can model many classical industrial problems such as capital budgeting, cargo loading and stock cutting (Martello and Toth, 1990).

This chapter presents several adaptations of Estimation of Distribution Algorithms (EDAs) for the knapsack problem. These adaptations differ in the representation that they used, the way in which they obtain their initial populations and the manner in which they treat the constraint related to the knapsack capacity. Experimental results obtained for problems of different sizes are used to compare different approaches based on EDAs.

The rest of the chapter is structured in the following way. Section 2 introduces the mathematical notation for the knapsack problem. In Section 3 a binary representation for the problem is presented, as the manner in which

P. Larrañaga et al. (eds.), *Estimation of Distribution Algorithms*

Table 9.1 0–1 knapsack problem with 7 items.

Item	1	2	3	4	5	6	7
Profit	20	31	17	30	14	52	10
Weight	30	54	32	16	27	61	7

discrete and continuous EDAs can be applied to it. The next section has the same structure, but the representation used is based on permutations. Section 5 presents experimental results, while conclusions are given in Section 6.

2. The 0–1 knapsack problem

The knapsack problem considered here is the 0–1 knapsack problem, which is classified as NP-hard (Garey and Johnson, 1979).

The 0–1 knapsack problem is: given a finite set of items where for each item its weight and profit are known, try to select the subset of items that provides the maximum profit, and whose sum of weights is bounded by the knapsack capacity.

In mathematical notation, if we denote by

- n the number of items
- p_i the profit of item i
- w_i the weight of item i
- c the capacity of the knapsack

then a solution for the 0–1 knapsack problem consists of selecting a subset of the items so as to:

- maximize $\sum_{i=1}^{n} p_i x_i$
- subject to the constraint $\sum_{i=1}^{n} w_i x_i \leq c$

where for all $i = 1, \ldots, n$

$$x_i = \begin{cases} 1 & \text{if item } i \text{ is selected} \\ 0 & \text{otherwise.} \end{cases}$$

Example 9.1 *We illustrate the 0–1 knapsack problem with a simple example consisting of 7 items, where the profits and weights associated with each item are shown in Table 9.1. We assume that the capacity of the knapsack is* $c = 100$.

If we select the items numbered $1, 2$ *and* 4 *we obtain a profit of* 81 *with a combined weight* $(30 + 54 + 16 = 100)$ *that doesn't exceed the capacity of the knapsack. However it is not possible to select the items numbered* $1, 2$ *and* 3 *because their combined weight* $(30 + 54 + 32 = 116)$ *exceeds the capacity of the knapsack.*

Approaches to the knapsack problem include both algorithms developed using greedy principles and exact methods. The greedy principle orders the items by nonincreasing efficiencies, where efficiency is the ratio between profit and weight, then includes the most efficient items in the knapsack until its capacity is exceeded. Approches based on the greedy principle include work by Ingargiola and Korsh (1973), Dembo and Hammer (1980), Martello and Toth (1988), Fayard and Plateau (1982) and Pisinger (1999). Alternatively, amongst the exact methods, Balas and Zemel's (1980) algorithm embeds the branch–and–bound technique, while Plateau and Elkihel's (1985) hybrid algorithm uses both branch–and-bound and dynamic programming.

More relevant to this chapter are the approaches based on Genetic Algorithms (GAs) and EDAs. Using GAs, Watannabe et al. (1992) and Gordon et al. (1993) developed an approach that uses binary representation, while Hinterding (1994) uses a representation based on permutations. Olsen (1994) proposes the use of penalty functions designed for the knapsack problem and Simoes and Costa (2001) propose an approach to the 0–1 knapsack problem based on GAs where the standard crossover operator is replaced by a biologically-inspired mechanism known as transposition.

Regarding EDAs, Baluja (1995) presents some results obtained with the PBIL algorithm, while Baluja and Davies (1998) show some empirical comparisons between COMIT and PBIL (see Chapter 3 for details of these algorithms). In both these works, binary representation is used.

3. Binary representation

In this section, we introduce two new approaches based on binary representations. The first is based on discrete EDAs, and the second uses EDAs in continuous domains. In both approaches we will assume that the variables will be ordered from left to right, by their ratios between profit and weight. This means that X_1 is the variable associated with the item with the largest ratio between profit and weight, and X_n corresponds to the variable with the worst ratio.

3.1 Discrete EDAs

Representation. Each possible solution to the 0–1 knapsack problem is represented by a binary array of dimension n, written as:

$$(x_1, \ldots, x_i, \ldots, x_n).$$

A value of 1 in the i^{th} position indicates that the i^{th} item has been selected for inclusion in the knapsack. From the point of view of the EDAs, each bit represents the value of one random variable following a Bernouilli distribution. The cardinality of the search space is 2^n.

Example 9.2 *Continuing the example introduced in Section 9.2, if the* 1^{st}, 2^{nd}, *and* 4^{th} *are the only selected items, then the corresponding binary array is:*

$$(1,1,0,1,0,0,0).$$

Evaluation. Since the array of bits can represent a solution that exceeds the capacity of the knapsack, we have developed two approaches to evaluate these arrays:

- *Penalization* of arrays representing non-feasible solutions.

  In this approach, if the array represents a non-feasible selection of items, we penalize its evaluation so that it is not competitive with the evaluations of feasible solutions.

  This evaluation is done in the following manner:

$$h(x_1,\ldots,x_i,\ldots,x_n) = \begin{cases} \sum_{i=1}^{n} p_i x_i & \text{if } \sum_{i=1}^{n} w_i x_i \leq c \\ K(\sum_{i=1}^{n} w_i - \sum_{i=1}^{n} w_i x_i) & \text{if } \sum_{i=1}^{n} w_i x_i > c \end{cases} \tag{9.1}$$

  where K is a positive number so that:

  for all $(x_1,\ldots,x_i,\ldots,x_n)$ such that $\sum_{i=1}^{n} w_i x_i > c$ we have:

$$K(\sum_{i=1}^{n} w_i - \sum_{i=1}^{n} w_i x_i) \leq \min(p_1,\ldots,p_n). \tag{9.2}$$

  Inequality 9.2 means that all the item selections that correspond to non-feasible solutions will obtain a worse evaluation than the evaluation corresponding to any feasible solution.

- *First fit* algorithm.

  In this approach to selecting items for inclusion in the knapsack whilst avoiding violating the constraint on its capacity, we select, from left to right and in order, items that meet the capacity constraint and reject items that result in a constraint violation. This first fit algorithm is shown in Figure 9.1. Hinterding (1994) uses this algorithm with different orderings to initialize the population of a GA.

Repeat

Search left to right for first item that does not violate the capacity constraint

if item found add to knapsack

else terminate algorithm

Figure 9.1 First fit algorithm.

Initialization. We have considered three different methods for obtaining the initial population:

- Each item is selected with equal probability, independent of the remaining items and independent of its ratio between profit and weight. This initialization will be called *uniform.*

  In order to obtain an expected number of selected items of $\frac{nc}{\sum_{i=1}^{n} w_i}$, each item is selected with a probability equal to $\frac{c}{\sum_{i=1}^{n} w_i}$.

  The probability vector from which we generate the initial population is therefore:

$$(p_0(x_1), \ldots, p_0(x_i), \ldots, p_0(x_n)) = (\frac{c}{\sum_{i=1}^{n} w_i}, \ldots, \frac{c}{\sum_{i=1}^{n} w_i}, \ldots, \frac{c}{\sum_{i=1}^{n} w_i}). \tag{9.3}$$

- Each item is selected with a probability *proportional* to its ratio between profit and weight. That is:

$$(p_0(x_1), \ldots, p_0(x_i), \ldots, p_0(x_n)) \propto (\frac{p_1}{w_1}, \ldots, \frac{p_i}{w_i}, \ldots, \frac{p_n}{w_n}). \tag{9.4}$$

  Denoting by M the proportionality constant and taking into account that the expected number of selected items must be $\frac{nc}{\sum_{i=1}^{n} w_i}$, we obtain that:

$$M \sum_{i=1}^{n} \frac{p_i}{w_i} = \frac{nc}{\sum_{j=1}^{n} w_j} \tag{9.5}$$

  or equivalently:

$$M = \frac{nc}{\sum_{j=1}^{n} w_j \sum_{i=1}^{n} \frac{p_i}{w_i}}, \tag{9.6}$$

obtaining finally that:

$$(p_0(x_1), \ldots, p_0(x_n)) =$$

$$\left(\frac{p_1 nc}{w_1 \sum_{j=1}^{n} w_j \sum_{i=1}^{n} \frac{p_i}{w_i}}, \ldots, \frac{p_n nc}{w_n \sum_{j=1}^{n} w_j \sum_{i=1}^{n} \frac{p_i}{w_i}} \right). \tag{9.7}$$

Since it is not guaranteed that each of the components of this vector is smaller than 1, we obtain the initial population of individuals using the following probability distribution:

$$p_0(x_i) = \min(1, \frac{p_i nc}{w_i \sum_{j=1}^{n} w_j \sum_{i=1}^{n} \frac{p_i}{w_i}}) \tag{9.8}$$

for all $i = 1, \ldots, n$.

- *Probabilistic seed*, in which starting from the solution provided by the first fit algorithm, an initial population of solutions is obtained by simulating the following probability distribution:

$$(p_0(x_1), \ldots, p_0(x_i), \ldots, p_0(x_n))$$

where for all $i = 1, \ldots, n$:

$$p_0(x_i) = \begin{cases} \alpha & \text{if item } x_i \text{ is selected by the first fit} \\ 1 - \alpha & \text{if item } x_i \text{ is not selected by the first fit.} \end{cases} \tag{9.9}$$

In this chapter we fix the α value to 0.95.

3.2 Continuous EDAs

Representation. For continuous EDAs, we need $n+1$ variables to represent each item selection. A Gaussian network is used to express the interdependencies between these $n + 1$ variables. The first n variables are related to their corresponding items, and the $(n+1)^{th}$ variable provides a threshold were chosen items have an associated variable which is larger than the threshold.

Example 9.3 *Suppose we use the following vector of dimension* 8 *to represent a choice between the 7 items in Example 9.1:*

$$(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) = (10.4, 12.8, 9.4, 16.2, 7.14, 5.67, 9.14, 9.98).$$

This array is interpreted here as selecting items 1, 2 *and* 4, *because their corresponding values -(10.4, 12.8 and 16.2)- are the only ones that overcomes the threshold of 9.98. This* 8 *dimensional array is therefore equivalent to the following* 7 *dimensional binary array:*

$$(1, 1, 0, 1, 0, 0, 0).$$

Evaluation. Evaluation is done on the binary transformation, so it is exactly the same as the one used for discrete EDAs.

Initialization. We again consider three different initializations:

- Each item is selected with *uniform* probability, independent of the remaining items and independent of its ratio between profit and weight.

  In a similar manner to the discrete case, if the probability of selecting each item is $\frac{c}{\sum_{i=1}^{n} w_i}$, and if $X_i \equiv \mathcal{N}(\mu, \sigma^2)$ for all $i = 1, \ldots, n$ and $X_{n+1} \equiv \mathcal{N}(\mu_{thr}, \sigma^2)$, we can obtain the value for μ_{thr}, given that:

$$p(X_i > X_{n+1}) = \frac{c}{\sum_{i=1}^{n} w_i} \tag{9.10}$$

  for all $i = 1, \ldots, n$. Noting that: $X_i - X_{n+1} \equiv \mathcal{N}(\mu - \mu_{thr}, 2\sigma^2)$ we obtain that the parameter μ_{thr} must satisfy:

$$p(X_i > X_{n+1}) = p(X_i - X_{n+1} > 0) = \int_{\mu_{thr}}^{\infty} \frac{1}{\sqrt{\pi} 2\sigma} e^{-\frac{1}{4\sigma^2}(x_i - \mu + \mu_{thr})^2} dx_i = \frac{c}{\sum_{i=1}^{n} w_i}. \tag{9.11}$$

  One example of this condition would be to use: $\sigma = \frac{1}{\sqrt{2}}$ and $\mu = \mu_{thr}$.

- Each item is selected with a probability *proportional* to its ratio between profit and weight.

  Reasoning in a similar way to the discrete case, we have that for all $i = 1, \ldots, n$:

$$p(X_i > X_{n+1}) = \frac{p_i n c}{w_i \sum_{j=1}^{n} w_j \sum_{i=1}^{n} \frac{p_i}{w_i}}. \tag{9.12}$$

  If we fix the parameters of the $(n+1)^{th}$ variable, $X_{n+1} \equiv \mathcal{N}(\mu_{thr}, \sigma^2)$ then each variable X_i $(i = 1, \ldots, n)$ which also follows a normal distribution, $X_i \equiv \mathcal{N}(\mu_i, \sigma^2)$, must satisfy:

$$p(X_i > X_{n+1}) = p(X_i - X_{n+1} > 0) = p(\mathcal{N}(\mu - \mu_{thr}, 2\sigma^2) > 0) =$$

$$\int_{0}^{\infty} \frac{1}{\sqrt{\pi} 2\sigma} e^{-\frac{1}{4\sigma^2}(x_i - \mu + \mu_{thr})^2} dx_i = \frac{p_i n c}{w_i \sum_{i=1}^{n} w_i \sum_{i=1}^{n} \frac{p_i}{w_i}}. \tag{9.13}$$

  In order to determine μ_i for all $i = 1, \ldots, n$ we fix the values for the parameters σ and μ_{thr} respectively to $\frac{1}{\sqrt{2}}$ and 0, to obtain that for all $i = 1, \ldots, n$:

$$p(\mathcal{N}(0,1) > 0) = \frac{p_i n c}{w_i \sum_{j=1}^{n} w_j \sum_{i=1}^{n} \frac{p_i}{w_i}}. \tag{9.14}$$

- *Probabilistic seed* as described in the previous section.

4. Representation based on permutations

Representation. Each possible solution for the 0–1 knapsack problem is represented by a permutation $(\pi(1), \ldots, \pi(i), \ldots \pi(n))$ of the items be selected.

Existing work on discrete EDAs (Santana et al., 1999; Bengoetxea et al., 2000a, 2000b) already deals with problems similar to the one of obtaining these permutations. These all adapt the simulation phase in order to obtain a permutation. The problem with these approaches is that the probability distribution learnt by the Bayesian network is changed by the constraint.

In this chapter we obtain each permutation from the simulation of a Gaussian network (Pelikan, 2000). We assume that the random variables in the Gaussian network are ordered –as in the binary representation– by their ratio between profit and weight.

If we denote by $(x_1, \ldots, x_i, \ldots, x_n)$ the continuous vector obtained in the simulation of the Gaussian network, then once the values x_i $(i = 1, \ldots, n)$ are ordered from the largest to the smallest we obtain the items: $\pi(i) = x_{i:n}$ for all $i = 1, \ldots, n$.

With this representation the cardinality of the search space is $n!$. This number is bigger (if $n \geq 4$) than 2^n because the representation we are using is redundant.

Example 9.4 *Assume that we have obtained the following 7 dimensional vector for the 7 items of Example 9.1:*

$$(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = (10.4, 12.8, 9.4, 16.2, 7.14, 5.67, 9.14).$$

Ordering the values corresponding to the items, we obtain:

$$(\pi(1), \pi(2), \pi(3), \pi(4), \pi(5), \pi(6), \pi(7)) = (3, 2, 4, 1, 6, 7, 5).$$

This permutation indicates the order of selection for the items to be included in the knapsack.

Evaluation. Here, we don't use the evaluation via penalization, so each permutation is evaluated using the first fit algorithm described in Section 9.3.

Initialization. We consider three possible initializations, as seen in Section 3:

- Each item has the same probability of being in each of the n positions of the permutation.

  To obtain this initialization all the random variables will follow the same normal distribution model. That is $X_i \equiv \mathcal{N}(\mu, \sigma^2)$ for all $i = 1, \ldots, n$.

- We assign more probability to those items with larger ratios between profit and weight.

Table 9.2 Knapsack problem. Binary representation. Average of the best results. $n = 50$. Greedy: 1713.

	penalty			*first fit*		
	uniform	*proportional*	*prob. seed*	*uniform*	*proportional*	*prob. seed*
UMDA	1731.8	1731.0	1717.8	1734.0	1734.0	1732.8
MIMIC	1731.2	1731.2	1716.8	1734.0	1734.0	1730.4
$EBNA_{PC}$	1731.0	1730.6	1720.4	1734.0	1733.0	1728.8
$UMDA_c$	1731.2	1733.2	1713.0	1734.0	1732.4	1713.0
$MIMIC_c$	1731.2	1732.2	1713.0	1734.0	1734.0	1713.0
$EGNA_{ee}$	1729.3	1730.7	1715.2	1734.0	1733.6	1717.6

Table 9.3 Knapsack problem. Binary representation. Average of the best results. $n = 200$. Greedy: 8010.

	penalty			*first fit*		
	uniform	*proportional*	*prob. seed*	*uniform*	*proportional*	*prob. seed*
UMDA	7964.0	7977.3	8011.6	8018.0	8018.0	8018.2
MIMIC	7977.2	7990.2	8013.0	8017.2	8018.0	8018.6
$UMDA_c$	7935.4	8003.8	8010.4	8016.8	8016.8	8014.5
$MIMIC_c$	7950.2	7985.0	8010.0	8017.8	8016.8	8014.1

Here, we generate n dimensional vectors whose i^{th} component have expected value proportional to its ratio between profit and weight. That is: $X_i \equiv \mathcal{N}(\mu_i, \sigma^2)$ where $\mu_i \propto \frac{p_i}{w_i}$ for all $i = 1, \ldots, n$.

- *Probabilistic seed*, as described in previous sections.

5. Experimental results

In this section we present the results of some experiments carried out with different number of objects ($n = 50, 200$ and 1000). For each experiment we randomly obtain the values for the profit and weight associated to each item, as well as the capacity of the knapsack.

In Tables 9.2 to 9.4 the average results over 10 independent runs for, respectively, the 50, 200 and 1000 objects problems are shown. All these three tables correspond to the results obtained with a binary representation. As can be seen in the tables we consider –see Section 3.1 for details– two ways for the evaluations of the individuals (penalization and first fit algorithm) in com-

Table 9.4 Knapsack problem. Binary representation. Average of the best results. $n = 1000$. Greedy: 41425.

	penalty			*first fit*		
	uniform	*proportional*	*prob. seed*	*uniform*	*proportional*	*prob. seed*
UMDA	38212.8	39063.6	40895.6	41145.6	41097.2	41393.0
MIMIC	38307.8	39282.8	41070.2	41341.8	41239.6	41424.4
$UMDA_c$	37545.8	40786.2	41425.0	41425.6	41425.0	41425.0
$MIMIC_c$	37647.2	39787.7	41425.0	41426.2	41425.4	41425.0

Table 9.5 Knapsack problem. Representation based on permutation. Mean of the best results. $n = 50$. Greedy: 1713.

	uniform	*proportional*	*probabilistic seed*
$UMDA_c$	1734.0	1734.0	1713.0
$MIMIC_c$	1734.0	1734.0	1713.0
$EGNA_{ee}$	1734.0	1733.6	1713.0

Table 9.6 Knapsack problem. Representation based on permutation. Mean of the best results. $n = 200$. Greedy: 8010.

	uniform	*proportional*	*probabilistic seed*
$UMDA_c$	8012.0	8012.1	8016.5
$MIMIC_c$	8005.8	8014.4	8016.2

bination with three initializations (uniform, proportional and by means of a probabilistic seeding).

In a similar manner Tables 9.5 to 9.7 present the results obtained for the permutation based representation –see Section 3.2 for details– for the 50, 200 and 1000 objects problems. In these tables we take into account the same three different initializations, but in these cases always the first fit algorithm was the only evaluation method considered.

Roughly speaking the best results were obtained with the first fit algorithm as the way to verify the constraints of the 0–1 knapsack problem. For the smallest problem considered ($n = 50$) the best results were obtained with the

Table 9.7 Knapsack problem. Representation based on permutation. Mean of the best results. $n = 1000$. Greedy: 41425.

	uniform	*proportional*	*probabilistic seed*
$UMDA_c$	40246.0	40585.0	41427.0
$MIMIC_c$	40246.8	40478.7	41427.0

uniform initialization and binary representation. In the intermediate problem ($n = 200$) the best results were obtained with the first fit evaluation and the binary representation in conjunction with discrete UMDA. Finally in the biggest problem ($n = 1000$), the first fit evaluation in combination with a permutation based representation and a probabilistic seeding led to the best results.

The non-parametric tests of Kruskal-Wallis and Man-Whitney were used to verify the null hypothesis of the same distribution. These tasks were carried out with the statistical package S.P.S.S. release 10.0.6. The results were as follows:

- Comparing different EDA algorithms.

  Here, fixing the representation (binary or permutation based), the evaluation (penalty or first fit) and the initialization type (uniform, proportional or probabilistic seeding), we aim to compare the results obtained with the different EDA approaches.

  - 50 objects

    The differences were statistically significant for the case of discrete EDAs for a binary representation, with a penalty evaluation and a uniform initialization ($p = 0.006$), and also with a first fit evaluation and a probabilistic seeding ($p = 0.0291$). We also found statistically significant differences for the case of continuous EDAs, for a binary representation, a first fit evaluation and a probabilistic seeding ($p = 0.0403$). On the other hand, with a permutation based representation the tests did not detect that the differences between the three continuous EDAs were statistically significant.

  - 200 objects

    The following cases presented differences statistically significant for discrete EDAs: binary representation, with a penalty evaluation and an uniform initialization ($p = 0.009$) or a proportional initialization ($p = 0.0058$), and also binary representation with a first fit evaluation and an uniform initialization ($p = 0.0293$). In the case

of continuous EDAs the tests showed differences for: binary representation with a penalty evaluation and proportional initialization ($p = 0.0013$), and for permutation based representation, and uniform initialization ($p = 0.0086$).

- 1000 objects

  For discrete EDAs all the differences were statistically significant except for the case of a penalty evaluation in conjunction with an uniform initialization. For continuous EDAs we obtained differences for: binary representation with penalty evaluation and proportional initialization ($p = 0.0001$), and permutation based representation with proportional initialization ($p = 0.0227$).

- Comparing different evaluations.

  The objective in this point is to compare the behaviour of the algorithms once the initialization and the type of EDA were fixed. In fact, these comparisons are only valid for the results presented in Tables 9.2 to 9.4.

  - 50 objects

    In the case of discrete EDAs with a binary representation, the obtained differences between pairs of algorithms of the same complexity and same initialization were statistically significant. When comparing continuous EDAs with a binary representation the cases with differences statistically significant were UMDAs with uniform initialization ($p = 0.0293$), MIMICs with uniform initialization ($p = 0.0049$), and proportional initialization ($p = 0.0049$) as well as EGNA$_{ee}$s with uniform initialization ($p = 0.0019$).

  - 200 objects

    In this case, for discrete EDAs as well as for continuous EDAs with a binary representation all the differences between pairs of algorithms of the same complexity and same initialization were statistically significant.

  - 1000 objects

    In this case we obtained the same behaviour as in the case of 200 objects, except for the continuous EDAs where the differences when comparing the two types of evaluations were not statistically significant for the probabilistic seeding based initializations.

- Comparing different representations.

  For the penalty evaluation we compare the results obtained with discrete and continuous EDAs of the same complexity and a binary representation.

For the first fit evaluation we extend the comparison taking into account permutation based representation.

- 50 objects

  The differences were significant for: UMDAs with penalty evaluation and proportional initialization ($p < 0.0001$), UMDAs with penalty evaluation and probabilistic seeding ($p = 0.0051$), UMDAs with first fit evaluation and proportional initialization ($p = 0.0115$), UMDAs with first fit evaluation and probabilistic seeding ($p < 0.0001$). Also MIMICs with first fit evaluation and probabilistic seeding ($p < 0.0001$) as well as EBNA_{PC} versus EGNA_{ee} with penalty evaluation and probabilistic seeding ($p = 0.0015$) and EBNA_{PC} versus EGNA_{ee} with binary representation versus EGNA_{ee} with continuous representation with first fit evaluation and probabilistic seeding ($p < 0.0001$) presented differences statistically significant.

- 200 objects

  In this example, all the differences were statistically significant except the following three cases: UMDAs with penalty evaluation and probabilistic seeding initialization ($p = 0.1351$), MIMICs with penalty evaluation and uniform initialization ($p = 0.1668$), and MIMIC with first fit evaluation and proportional initialization ($p = 0.3420$).

- 1000 objects

  In this example all the differences were statistically significant.

- Comparing different initializations.

  Here we compare, for algorithms with the same complexity and the same evaluation type, the results obtained for the three different initializations: uniform, proportional and probabilistic seeding.

  - 50 objects

    All the differences were statistically significant except for the case of UMDA algorithms with a binary representation, and first fit evaluation ($p = 0.1260$).

  - 200 objects

    Except for UMDA algorithms with binary representation, and first fit evaluation ($p = 0.4508$) all the differences were statistically significant.

  - 1000 objects

    In all the comparisons the obtained differences were statistically significant.

6. Conclusions

In this chapter we have introduced for the first time the application of EDAs to the 0–1 knapsack problem. We have introduced two different representations (binary and permutation based) in combination with two manner of maintaining the feasibility of the individuals (penalization and first fit algorithm) and also three different initializations of the first population (uniform, proportional and probabilistic seeding).

With the experiment we have carried out in this preliminary work, we conclude the superiority of the first fit algorithm with respect to the penalization. More work must be done to obtain clear conclusions with respect to the other parameters.

References

Balas, E. and Zemel, E. (1980). An algorithm for large zero–one knapsack problems. *Operations Research*, 28:1130–1154.

Baluja, S. (1995). An empirical comparison of seven iterative and evolutionary function optimization heuristics. Technical report, School of Computer Science. Carnegie Mellon University. CMU-CS-95-193.

Baluja, S. and Davies, S. (1998). Fast probabilistic modeling for combinatorial optimization. In *AAAI-98*.

Bengoetxea, E., Larrañaga, P., Bloch, I., Perchant, A., and Boeres, C. (2000). Inexact graph matching using learning and simulation of Bayesian networks. An empirical comparison between different approaches with synthetic data. In *Workshop Notes of CaNew2000: Workshop on Bayesian and Causal Networks: From Inference to Data Mining. Fourteenth European Conference on Artificial Intelligence, ECAI2000.*

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

Dembo, R.S. and Hammer, P.L. (1980). A reduction algorithm for knapsack problems. *Methods of Operational Research*, 36:49–60.

Fayard, D. and Plateau, G. (1982). An algorithm for the solution of the 0–1 knapsack problem. *Computing*, 28:269–287.

Garey, M.R. and Johnson, D.S. (1979). *Computers and Intractability. A Guide to the Theory of NP–completeness.* W.H. Freman Co., San Francisco.

Gordon, W.S., Bohm, A.P.W., and Whitney, D. (1993). A note on the performance of genetic algorithms on zero–one knapsack problem. Technical report, Department of Computer Science. Technical Report CS-93-108. Colorado State University.

Hinterding, R. (1994). Mapping, order–independent genes and the knapsack problem. In *IEEE Conference*, pages 13–17.

Ingargiola, G.P. and Korsh, J.F. (1973). A reduction algorithm for zero–one single knapsack problems. *Management Science*, 20:460–463.

Martello, S. and Toth, P. (1988). A new algorithm for the 0–1 knapsack problem. *Management Science*, 34:633–644.

Martello, S. and Toth, P. (1990). *Knapsack Problems: Algorithms and Computer Implementations.* John Wiley and Sons.

Olsen, A.L. (1994). Penalty functions and the knapsack problem. In *IEEE Conference*, pages 554–558.

Pelikan, M. (2000). Solving permutation problems with continuous EDAs. *Personal communication.*

Pisinger, D. (1999). Core problems knapsack algorithms. *Operations Research*, 47(4):570–575.

Plateu, G. and Elkihel, M. (1985). A hybrid algorithm for the 0–1 knapsack problem. *Methods of Operations Research*, 49:277–293.

Santana, R. and Ochoa, A. (1999). Dealing with constraints with Estimation of Distribution Algorithms: The univariate case. In *Second Symposium on Artificial Intelligence. Adaptive Systems. CIMAF 99*, pages 378–384.

Simoes, A. and Costa, E. (2001). An evolutionary approach to the zero–one knapsack problem: Testing ideas from biology. In Kurková, V., Steel, N. C., Neruda, R., and Kárný, M., editors, *International Conference on Artificial Neural Networks and Genetic Algorithms. ICANNGA-2001*, pages 236–239. Springer.

Watannabe, K., Ikeda, Y., Matsuo, S., and Tsuji, T. (1992). Improvements of the genetic algorithms and its applications. Technical report, Faculty of Engineering Fuki University. Vol. 40, Issue 1.