A Comparison of Differential Evolution Algorithm with Binary and Continuous Encoding for the MKP

Jonas Krause
Technological Federal University of Paraná (UTFPR)
Laboratory of Bioinformatics
Curitiba, Paraná, Brazil
Email: jkfries@gmail.com

Heitor Silvério Lopes
Technological Federal University of Paraná (UTFPR)
Laboratory of Bioinformatics
Curitiba, Paraná, Brazil
Email: hslopes@utfpr.edu.br

***Abstract*—This paper provides a brief description on how continuous algorithms can be applied to binary problems. Differential Evolution is the continuous algorithm studied and two versions of this algorithm are presented: the Binary Differential Evolution with a binary encoding and the Discretized Differential Evolution with a continuous encoding. Several discretization methods are presented and the most used method in literature is implemented for the solution discretization. Benchmarks with different complexity and search space sizes of the 0-1 Multiple Knapsack Problem are used to compare the performance of each Differential Evolution algorithm presented and the Genetic Algorithm with binary encoding. Results suggest that continuous methods can be very efficient when discretized for binary spaces.**

I. Introduction

The discrete optimization problems have benefited with the use of continuous strategies. The 0-1 Multiple Knapsack Problem is one of them and this paper aim to demonstrate that continuous algorithms can be applied in combinatorial problems efficiently. In literature, algorithms designed for continuous search spaces have been successfully applied in discrete optimization problems. However, there is no final conclusion on what is the best strategy to apply these algorithms in combinatorial problems.

In 2006, [1] presented a Binary Differential Evolution using an angle modulation. It consists of a generation of a bit string using a trigonometric generating function. One year later [2] also implemented the angle modulation strategy for another Binary Differential Evolution version. Other techniques are used by [3], like the homomorphous mapping and the interpretation of the continuous solution vector as a vector of probabilities. In 2008, [4] presented a self-adaptive Differential Evolution algorithm, a rank-based representation schema is used to transform the discrete combinatorial problem into a continuous domain and it only produces feasible solutions. Another Binary Differential Evolution was presented by [5] and this algorithm uses a mapping operator to convert the discrete variables into continuous ones, a sigmoid function for the mutation operator and an inverse mapping operator to transform the continuous variables back to discrete. The mutation operator was also the focus of [6] and [7]. Both used a sigmoid function to discretize the mutation process. Recently, the strategy of using a mixed discrete-continuous optimization has also been applied to the Differential Evolution algorithm [8]. Other discretization methods are described in [9] and most of them can be applied to Differential Evolution.

A classical binary optimization problem is used to test the above mentioned algorithms, the 0-1 Multiple Knapsack Problem. This is one of the most used variants of the Knapsack Problem [10], [11] and it is used to optimize logistic, transportation and production activities [12]. Algorithms using the solution vector with binary variables can represent the best combination of n different items to be carried in m different knapsacks.

This paper presents two different methods to apply the continuous Differential Evolution to binary problems. The first method consists in using only binary variables, ensuring that only valid solutions will be created. For this method, the mutation process of the Differential Evolution was replaced by a Genetic Algorithm inspired mutation. Instead of using the weighted differences of population members, the mutation process is a bit inversion. Consequently, the optimization process will handle only binary variables. The second method discretizes the solution vector, preserving all the characteristics of Differential Evolution. Therefore, the algorithm will continue to handle continuous variables and search for solutions in the continuous space.

The objective of this paper is to present discretization techniques that can be used in the continuous Differential Evolution algorithm and support the good applicability of continue devised methods in combinatorial problems. The results of this discretized algorithm are compared with two binary encoding algorithms, the Genetic Algorithm and a binary version of the Differential Evolution. Despite the several ways of discretization, the capability of maintaining the continuous characteristics of the algorithm may be the key to efficiently apply continuous methods in binary and combinatorial problems.

II. The Knapsack Problem

The Knapsack Problem (KP) consists in selecting the most profitable combination of items to be carried in a knapsack. Each item has associated profit and restriction values and the restrictions total should not exceed the knapsack capacity.

The KP is part of a class problems called NP-complete, thus the possible solutions increase exponentially and it cannot be solved in polynomial-time. It is presented in literature in several variants [11], the 0-1 Knapsack, the Bounded Knapsack, the Subset-Sum, the Change-Making, the Generalized Assignment, the Bin-Packing and the 0-1 Multiple Knapsack.

A. The 0-1 Multiple Knapsack Problem

The 0-1 Multiple Knapsack Problem (MKP) is the classic and most complete variant of the KP. Each item n has a profit value and different restrictions for each knapsack. The binary variables X_i $(i = 1, 2, ..., n)$ represent whether the n-th item will be part of the solution (1) or not (0). The objective is to find the best combination of X_i items that maximize the sum of the profit P_i, represented in Equation 1.

$$max(\sum_{i=1}^{n}(P_i.X_i)) \tag{1}$$

However, each knapsack m has a maximum capacity C_j $(j = 1, 2, ..., m)$ and the sum of the restrictions W_{ij} should be less or equal than each capacity for all j. This restriction function is represented in Equation 2.

$$\sum_{i=1}^{n}(W_{ij}.X_i) \leq C_j, \forall j \tag{2}$$

The total number of possible combinations of the binary variable X_i will depend on how many items will be carried and the number m of knapsacks. The size of the Search Space (SS) created by these two variables is the total number of possible solutions and they increase exponentially according to n, represented in Equation 3.

$$SS = m.2^n \tag{3}$$

The SS is also modeled by the restrictions and the knapsack capacities, creating more complex search spaces. Therefore, deterministic algorithms can take excessive time to find the best solution as the number n of items increases. As alternative, meta-heuristic methods can be used to find the best solutions evolving an initial random population.

III. Discretization Methods

To apply continuous algorithms in combinatorial, binary and categorial (discrete) problems, it is necessary to reduce the total number of feasible solutions. The discretization methods adapt the continuous devised algorithms to discrete problems [9]. The most simple adaptation is to restrict the variables to discrete only, turning the Search Space (SS) to a discrete one. This discretization method changes the original encoding of the algorithm and it will generate only feasible solutions. The Binary Differential Evolution (BDE) presented in this paper uses this technique. It restricts the variables to binary, thus searching in binary spaces only.

Other discretization methods allow continuous variables and restrict some specific steps of the algorithm. The angle modulation [1], the homomorphous mapping and the interpretation of the continuous solution vector as a vector of probabilities [3] are discretization methods which have already been presented. The rank-based representation schema [4], the mapping operator and the sigmoid function to discretize the mutation process also appear in literature [5]. These techniques create new opportunities to apply the Differential Evolution (DE) to discrete problems.

A recent survey [9] presents other discretization methods that can be applied to the DE algorithm as well as to other similar continuous meta-heuristics. As result of this survey, the strategy of use a Sigmoid Function applied in one step of the algorithm appeared as the most frequent and efficient discretization technique. The Random-Key discretization method is the second most used technique and it consists in setting the continuous solution in ascending order to encode it in a discrete order. The Nearest Integer method converts the continuous value to the nearest integer by truncating it up or down. The Smallest Position Value maps the continuous solution by placing the index of the lowest value component as the first item on a permutated solution, the next lowest as the second and so on. And the Great Value Priority method uses the same technique, but it sorts the solution by the largest element. The Discretized Differential Evolution (DDE) presented in this paper use a sigmoid function to discretize a normalized solution vector to form a bit string.

IV. Genetic Algorithm

The Genetic Algorithm (GA) is a heuristic optimization method presented by [13] and it has been used for high complexity problems. Based on the Darwin's evolution theory, this population algorithm evolves the individuals solutions through a predetermined number of generations. Each individual is represented by its chromosome, a solution vector that can be encoded as binary, integer or real numbers. These characteristics made the GA one of the most used algorithms for binary and combinatorial problems.

For the MKP, the GA is applied with a binary chromosome. An initial random population is created and each individual has n dimensions, representing each item of the MKP. A trial population is created by using a selection method, stochastically selecting individuals from one generation to create the basis for the next generation. The crossover process represents the reproduction of the population, two individuals are selected and their chromosomes are crossed to create a child chromosome. The mutation process consists in inverting a random bit of the chromosome creating a new individual. This new population is evaluated as per each individual fitness, generally associated with the objective function of the problem.

In this discrete optimization method, population evolutionary techniques are used to achieve the best chromosome. The best binary combination of the solution vector is the most evolved individual. This evolution occurs in a binary space and the decision making criterion to change the dimension of the n-th element from 0 to 1 or vice-versa is abrupt.

V. Differential Evolution

The Differential Evolution (DE) algorithm was first introduced by [14] and devised for optimization problems in continuous spaces. It arose as a simple and efficient heuristic for global optimization.

DE is a population based algorithm and has different strategies. The most widely used and successful strategy is the DE/rand/1/bin [15]. This strategy consists in selecting a random individual to be mutated, a difference of random vectors to perturb the individual selected and a binomial crossover. The mutation and crossover processes are applied in a trial population, each individual fitness is calculated and the new population is evaluated. Using continuous variables, the

DE searches the most evolved individual for a predetermined number of generations.

The formulation of the DE/rand/1/bin strategy is shown in Equation 4:

$$v(g,i,j) = x(g,r_3,j) + F \times [x(g,r_1,j) - x(g,r_2,j)] \quad (4)$$

This equation represents the trial vector v receiving the random vector x plus a random difference variation. The DE/**rand**/1/bin represents the random individual x to be mutated. The parameter F is the weighting factor used to control the amplification of the differential variation. This strategy may vary depending on the individual selected, an elitism can select the best individual for the mutation process and the strategy would be DE/**best**/1/bin. The best individual can also be added on the differential variation, in this case the strategy is called DE/**rand-to-best**/1/bin. Other strategies consist in using two differential variations, represented by DE/rand/**2**/bin, DE/best/**2**/bin and DE/rand-to-best/**2**/bin. The last element is the crossover process and it can be set to DE/rand/1/**bin** and DE/rand/1/**exp**, representing the binomial and exponential crossovers respectively. The exponential crossover process can also be used in all of the described methods, creating a list of ten possible strategies that can be used on DE [16]. Both proposed methods in this paper use the classical DE/rand/1/bin.

A. Binary Differential Evolution

This version of the Binary Differential Evolution (BDE) was presented in [10] and it is adapted for binary spaces only. The adaptation of the original DE starts on the initialization of the population, instead of initiating with random continuous values the individuals are initiated with random binary values. The original mutation process of DE is replaced by a random bit inversion. This adaptation of the DE mutation process is inspired on the GA. The perturbation process is a new parameter inserted to establish how many individuals of the population will pass through the mutation and crossover processes. This new parameter also ensures that at least one individual will be mutated. The DE crossover process is kept as the original since all individuals will continue to be binary after it. This process is activated during the perturbation procedure and after the mutation.

Figure 1 represents the pseudocode of the BDE algorithm.

The BDE algorithm initializes with the parameters settings. The first parameter is the individual dimension or $range$ and for the 0-1 MKP it represents the number n of items. NP is the number of individuals of the population. The PM parameter is the mutation rate and the PR parameter is the permutation rate. A random population $\vec{x_i}$ is created and the fitness of each individual is calculated $f(\vec{x_i})$. Each individual is selected randomly by the permutation rate and it is submitted to the mutation and crossover processes. The new individual $\vec{y}$ has its fitness calculated $f(\vec{y})$. If the new fitness of the trial individual is better than the previous individual fitness $f(\vec{x_i})$, the trial solution $\vec{y}$ will be part of the new population.

Due the fact that the algorithm is restricted to 0 or 1 variables, the sign function may represent the decision making criteria of each dimension of the individual. In this scenario, the dimensions also do not change gradually from 0 to 1 or vice-versa.

Fig. 1. Binary Differential Evolution (BDE)

```
Parameters: range, NP, PM, PR
Initial Population x_i (i = 1, ..., range)
Evaluate fitness f(x_i) of each individual
while not done do
  for i = 1 to NP do
    if rndreal(0,1) < PR ou j = j_rand then
      if rndreal(0,1) < PM then
        InverterBit(y_j)
      end if
      Crossover(y_j)
    end if
  end for
  Calculate fitness f(y)
  if f(y) > f(x_i) then
    x_i <== y
  end if
  Evaluate x*
end while
Print Results
```

B. Discretized Differential Evolution

This novel version of the Differential Evolution is a continuous encoded algorithm that discretizes only the solution vector. The individual to be evolved by the Discretized Differential Evolution (DDE) is a n dimensional vector and each dimension is populated with a random float number between -1 and 1. With a normalized entry for each individual of the initial population, the DDE proceeds with the DE/rand/1/bin strategy of mutation and crossover through the generations. The new population is a group of continuous variables vectors and these individuals have to be discretized to have their fitness evaluated. This discretization method uses a sigmoid function that allows each dimension to evolve gradually and individually.

Let F be the float number in each i dimension, the solution vector is discretized using the result of the sigmoid function of F. If it is greater than zero the i-th dimension is set to 1, otherwise, it is set to 0. This discretization process is represented by Equation 5.

$$X_i = \begin{cases} 1, \text{if } \frac{2}{1+\exp(-2.F_i)} - 1 > 0, \\ 0, \text{otherwise} \end{cases} \quad (5)$$

Using this strategy, the evolved dimensions do not jump from 0 to 1 and vice-versa in binary spaces. They evolve gradually around zero using weighted values to search for the best continuous combination. This characteristic may be the key to efficiently apply continuous algorithms to discrete problems.

Figure 2 presents the pseudocode of the BDE algorithm.

The DDE algorithm initializes with the parameters NP, CR, F and range. The NP states for population, the total number of individuals. The CR parameter is the crossover rate and the F is the mutation rate. The range is the dimension of each individual and for the MKP it is the number of items. An initial random population is created with NP individuals and their initial fitness is calculated. Through the number of

Fig. 2. Discretized Differential Evolution (DDE)

```
Function f(x) = DE (range, NP, CR, F)
x ⇐ random (range, NP)
fit_x ⇐ f(x)
while not done do
  for i = 1 to NP do
    v_{i,G+1} ⇐ mutation (x_{i,G}, F)
    u_{i,G+1} ⇐ crossover (x_{i,G}, v_{i,G+1}, CR)
  end for
  if sigmoid (u_{i,G+1}) > 0 then
    u_{i,G+1} ⇐ 1
  else
    u_{i,G+1} ⇐ 0
  end if
  fit_u ⇐ f(u)
  for i = 1 to NP do
    if fit_u(i) > fit_x(i) then
      x_{i,G+1} ⇐ u_{i,G+1}
    else
      x_{i,G+1} ⇐ x_{i,G}
    end if
  end for
end while
```

generations previously set, a trial population is created using the mutation and crossover processes. This new population is discretized by the sigmoid function which assigns the values 1 or 0, depending whether the continuous dimension of the individual is greater than 0 or not. The fitness of this trial and discretized population is calculated and if the trial individual fitness is greater than the previous one, the new individual is incorporated to the new population.

The same strategy used when applying the DDE to this binary problem can also be used to convert continuous values into integer values and, consequently, apply the DE to combinatorial problems. Hence, this version of the DE may be called discretized and can be adapted to other binary or combinatorial problems.

VI. Computational Results

The GA implementation was based on GALOPPS 3.2.4[1] (Genetic ALgorithm Optimized for Portability and Parallelism System). This software is a flexible and generic algorithm written in ANSI C and based on the Simple Genetic Algorithm (SGA) [17]. The BDE and the DDE algorithms were implemented using DE 3.6[2], also in ANSI C.

The benchmarks were carefully selected from OrLib [3,4] to represent the diverse search spaces (SS). They are presented on Table I and available in [18]. They are sorted by their SS sizes, but the structure of the SS also depends on the restrictions and the knapsack capacities to define its complexity. The benchmarks PB1 and WEING1 have the same SS size, but different complexity due to this fact. These characteristics help to understand the behavior of each algorithm when analysing the further results.

[1]http://garage.cse.msu.edu/software/galopps/

[2]http://www1.icsi.berkeley.edu/storn/code.html

[3]http://people.brunel.ac.uk/~ mastjjb/jeb/orlib/files/mknap2.txt

[4]http://www.cs.nott.ac.uk/~ jqd/

TABLE I. OrLib Benchmarks

Benchmark	n	m	SS
PB5	20	10	10×2^{20}
PB1	27	4	4×2^{27}
WEING1	28	2	2×2^{28}
PB4	29	2	2×2^{29}
WEISH1	30	5	5×2^{30}
PB2	34	4	4×2^{34}
PB7	37	30	30×2^{37}
PB6	40	30	30×2^{40}
WEISH14	60	5	5×2^{60}
SENTO1	60	30	30×2^{60}
GK1	100	15	15×2^{100}
WEING7	105	2	2×2^{105}
GK3	150	25	25×2^{150}

TABLE II. Parameters

Parameter	GA	BDE	DDE
Population	100	100	100
Generations	300	300	300
Mutation	0.05	0.05	0.05
Crossover	0.8	-	0.8
Perturbation	-	0.5	-

The benchmarks selected represent some of the most complex instances of the MKP with known optimum. The optimum value is useful for evaluating quality of solutions provided by each algorithm. Other instances with 250 and 500 knapsacks are available at the Orlib repository, but with no known optimum.

Since GA and DE are population-based algorithms, the total number of individuals to be evaluated and the number of generations have to be defined. Despite of having different mutation and crossover processes, their corresponding parameters were similarly adjusted to compare the algorithms results. The parameters used on each algorithm are selected as previous studies [9], [10] and listed on Table II.

Table III presents the benchmarks, the known optimum (Opt) and, for each method, the best solution found (Best), the percentage of the known optimum achieved by the best solution (% Opt), and the percentage of success (% Suc) that is how many times the optimum solution was found in all runs. The known optimum is the best possible binary combination for an individual and it is represented on Table III as the highest possible profit. Since GA, BDE and DDE are stochastic methods, each algorithm was run 100 times with different random seeds for the initial population. These results indicate the capability and robustness of each algorithm as well as the complexity of each benchmark. The Average (Avg) and the Standard Deviation (SD) are important information to be evaluated as well, they are presented on Table IV. The average represents how close to the known optimum all the solutions are and, consequently, it is related to the repeatability and reliability of the algorithm.

Analysing the data on Table IV, the DDE has the highest average in most benchmarks (PB5, PB1, WEING1, PB4, PB6, WEISH14, SENTO1, GK1, WEING7 and GK3). DDE also has the lowest standard deviation values in almost all benchmarks, showing the low dispersion that each best-of-run solution has from the average. These results suggest that the 100 DDE solutions are satisfactory, specially for the largest SS benchmarks (SENTO1, GK1, WEING7 and GK3).

To better understand the difference between the BDE and DDE and how this continuous algorithm works, the best binary

TABLE III. BEST INDIVIDUALS RESULTS

Benchmark	Opt	GA Best	GA % Opt	GA % Suc	BDE Best	BDE % Opt	BDE % Suc	DDE Best	DDE % Opt	DDE % Suc
PB5	2139	**2139**	0%	8%	**2139**	0%	64%	**2139**	0%	72%
PB1	3090	**3090**	0%	6%	**3090**	0%	28%	**3090**	0%	30%
WEING1	141278	**141278**	0%	34%	**141278**	0%	78%	**141278**	0%	98%
PB4	95168	**95168**	0%	11%	91544	3.81%	0%	**95168**	0%	80%
WEISH1	4554	**4554**	0%	28%	**4554**	0%	100%	**4554**	0%	95%
PB2	3186	**3186**	0%	23%	**3186**	0%	89%	**3186**	0%	68%
PB7	1035	1000	3.38%	0%	**1035**	0%	22%	**1035**	0%	76%
PB6	776	765	1.42%	0%	765	1.42%	0%	**776**	0%	40%
WEISH14	6954	6723	3.32%	0%	6914	0.58%	0%	**6954**	0%	100%
SENTO1	7772	7543	2.95%	0%	7596	2.26%	0%	7761	0.14%	0%
GK1	3766	3643	3.27%	0%	3696	1.86%	0%	3740	0.69%	0%
WEING7	1095445	1078722	1.53%	0%	1089841	0.51%	0%	1095206	0.02%	0%
GK3	5656	5470	3.29%	0%	5538	2.09%	0%	5603	0.94%	0%

TABLE IV. AVERAGE AND STANDARD DEVIATION RESULTS

Benchmark	GA Avg ± SD	BDE Avg ± SD	DDE Avg ± SD
PB5	2097.60 ± 24.60	2132.90 ± 8.20	**2133.72 ± 9.22**
PB1	3036.90 ± 27.20	3075.80 ± 13.0	**3079.12 ± 8.33**
WEING1	**141277.60 ± 2.80**	141277.40 ± 3.40	141268.00 ± 70.35
PB4	91711.67 ± 1421.59	88802.99 ± 718.08	**94716.92 ± 1168.02**
WEISH1	4544.40 ± 21.60	**4554.00 ± 0.00**	4552.88 ± 5.49
PB2	3150.82 ± 32.55	**3183.76 ± 8.41**	3180.16 ± 9.70
PB7	965.84 ± 21.64	**1033.01 ± 2.73**	1032.28 ± 4.96
PB6	723.81 ± 17.44	730.42 ± 27.46	**766.07 ± 10.59**
WEISH14	6427.90 ± 124.60	6795.80 ± 53.10	**6954.00 ± 0.00**
SENTO1	7094.00 ± 214.30	5395.50 ± 930.50	**7737.60 ± 23.92**
GK1	3612.60 ± 10.80	3693.20 ± 4.24	**3724.6 ± 8.38**
WEING7	1052993.90 ± 7279.30	1087073.10 ± 1285.90	**1093786.76 ± 900.26**
GK3	5426.16 ± 14.34	5534.95 ± 6.28	**5590.25 ± 11.04**

and continuous individuals found for the benchmark SENTO1 are compared in Figure 3. The binary individual from BDE and the continuous individual from DDE presented several similar dimensions after the DE discretization. In the continuous solution, the positive dimensions represent the items being carried on the knapsack and the negative dimensions the items carried off. For the total of 60 dimensions, only 2 different items are carried on and 2 items are carried off. After the discretization process, the continuous solution vector normalized between 1 and -1 will assume binary values. Figure 3 presents part of these individuals and highlights the difference between them for dimensions 37 and 41.

In Figure 3, BDE and DDE lines represent, respectively, part of discrete and continuous-valued solutions. Observing DDE, the decision making criterion leads to gradual changes in the items carried on/off the knapsack. The restrictions of each item and the capacity of each knapsack made the continuous dimensions float around zero. As example, the best continuous combination for the benchmark SENTO1 on item 41 has a positive value very near to zero. After the discretization the item 41 is carried on the knapsack by the continuous solution (DDE) but not on the discrete solution (BDE). These continuous dimensions solution resulted in a better binary combination after the individual discretization. This individual evolution combined with DE continuous strategies leaded to satisfactory solutions for the MKP.

To verify the relevance of the results of each method, Figure 4 shows the boxplot of the 100 solutions found for the benchmark GK3. This is one of the most complex benchmarks and with the largest SS. The boxplot represents the first and third quartiles, the median, the maximum and minimum.

The analysis of the Figure 4 shows the statistical non-

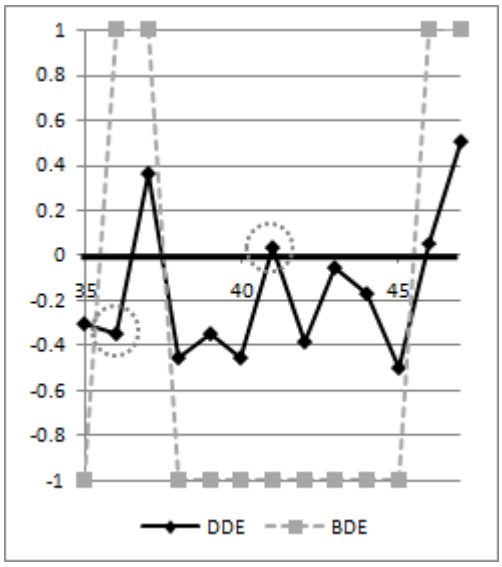

Fig. 3. Dimensions 35 to 47 of the best BDE and DDE individuals on benchmark SENTO1

overlapping intervals. Each interval represents the group of solutions achieved by each method and, consequently, they demonstrate different groups of solutions. Other two statistical tests were used; the Shapiro-Wilk, to verify the if the samples come from a normally distributed population and the Kruskal-Wallis, to verify if the results come from the same distribution. In the Shapiro-Wilk test the observed distributions nullify the test's primary hypothesis and, consequently, the analysed distribution is not a normal one. The Kruskal-Wallis test with a 5% of significance rejects the hypothesis that these results come from the same distribution. These statistical analyzes

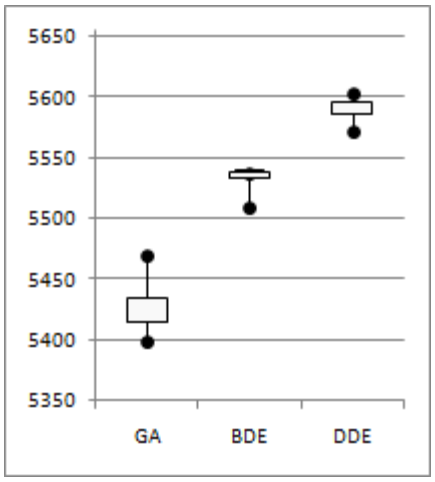

Fig. 4. BoxPlot of 100 results of each method for instance GK3

support the relevance of the results for each method and they present different groups of solutions.

The results presented in Table III (Best, % Opt and % Suc), Table IV (Avg and SD) and their statistical analysis point to a very robust and efficient algorithm. The DDE uses the continuous methods and techniques from DE to search for the continuous combination that will better fit the binary problem after the discretization procedure.

VII. Conclusion

The continuous algorithms devised to handle real numbers are frequently deployed for real-world optimization problems. Several discretization methods provide the possibility to apply these algorithms to discrete problems as well. Two versions of the DE are implemented to verify how this algorithm behaves using the binary and continuous encoding for the MKP benchmarks. The results pointed out an efficient discretized DE algorithm using a continuous encoding and a good percentage of success even being an stochastic method. Therefore, the continuous DE algorithm can be successfully applied to binary problems using a discretization method to convert the continuous solution to a binary one.

Future work using DDE includes the investigation of other discretization methods and different DE strategies to understand their advantages to improve the DDE performance. The adaptation of the DDE to combinatorial problems is also one of the main future goals. The satisfactory results in binary problems encourage the continuation of the research and reinforces the benefits of applying continuous algorithms to discrete problems.

Acknowledgment

The authors would like to thank Dr. Rafael Stubs Parpinelli (UDESC) for his contributions and the National Council for the Improvement of Higher Education (CAPES) for the scholarship to J. Krause. This work was also partially supported by a research grant from the CNPq to H. S. Lopes.

References

[1] G. Pampara, A. Engelbrecht, and N. Franken, "Binary differential evolution," in *IEEE Congress on Evolutionary Computation*, 2006, pp. 1873–1879.

[2] A. E. Kanlikilicer, A. Keles, and A. S. Uyar, "Experimental analysis of binary differential evolution in dynamic environments," in *Proceedings of the Conference on Genetic and Evolutionary Computation*. New York, NY, USA: ACM, 2007, pp. 2509–2514.

[3] A. P. Engelbrecht and G. Pampara, "Binary differential evolution strategies." in *IEEE Congress on Evolutionary Computation*, 2007, pp. 1942–1947.

[4] J. Zhang, V. Avasarala, A. C. Sanderson, and T. Mullen, "Differential evolution for discrete optimization: An experimental study on combinatorial auction problems." in *IEEE Congress on Evolutionary Computation*, 2008, pp. 2794–2800.

[5] C. Deng, B. Zhao, Y. Yang, and A. Deng, "Novel binary differential evolution algorithm for discrete optimization," in *Proceedings of the 2009 Fifth International Conference on Natural Computation*, vol. 4. Washington, DC, USA: IEEE Computer Society, 2009, pp. 346–349.

[6] L. Hou, H. Zhou, and J. Zhao, "A novel discrete differential evolution algorithm for stochastic VRPSPD," *Journal of Computational Information Systems*, vol. 6, pp. 2483–2491, 2010.

[7] G. Hou and X. Ma, "A novel binary differential evolution for discrete optimization," *Key Engineering Materials*, vol. 439-440, pp. 1493–1498, Jun. 2010.

[8] D. Lichtblau, "Differential evolution in discrete optimization," *International Journal of Swarm Intelligence and Evolutionary Computation*, vol. 1, pp. 1–10, Jun. 2012.

[9] J. Krause, J. A. Cordeiro, R. Parpinelli, and H. Lopes, "A survey of swarm algorithms applied to discrete optimization problems," in *Swarm Intelligence and Bio-inspired Computation: Theory and Applications*. Elsevier Science & Technology Books, 2013, pp. 169–191.

[10] J. Krause, R. Parpinelli, and H. Lopes, "Proposta de um algoritmo inspirado em evolucao diferencial aplicado ao problema multidimensional da mochila," in *Anais do Encontro Nacional de Inteligencia Artificial*. Curitiba, PR: SBC, oct 2012.

[11] S. Martello and P. Toth, *Knapsack problems: algorithms and computer implementations*. New York, USA: John Wiley & Sons, 1990.

[12] P. C. Chu and J. E. Beasley, "A genetic algorithm for the multidimensional knapsack problem," *Journal of Heuristics*, vol. 4, no. 1, pp. 63–86, 1998.

[13] J. H. Holland, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. Ann Arbor, USA: University of Michigan Press, 1975.

[14] R. Storn and K. Price, "Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces," Tech. Rep., 1995.

[15] P. C. B.V. Babu and J. Mubeen, "Multiobjective differential evolution (mode) for optimization of adiabatic styrene reactor," *Chemical Engineering Science*, vol. 60, no. 17, pp. 4822–4837, September 2005.

[16] J. Adeyemo and F. Otieno, "Differential evolution algorithm for solving multi-objective crop planning model," *Agricultural Water Management*, vol. 97, no. 6, pp. 848–856, June 2010.

[17] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA, USA: Addison-Wesley, 1989.

[18] J. E. Beasley, "OR-Library: distributing test problems by electronic mail," *Journal of the Operational Research Society*, vol. 41, no. 11, pp. 1069–1072, 1990.