ISSN (e): 2250 – 3005 || Volume, 14 || Issue, 4|| Jul. - Aug. – 2024 ||
International Journal of Computational Engineering Research (IJCER)

An Enhancement of Simple Differential Evolution Technique in Solving Knapsack Problem

Nibedita Chhatoi [1] Subhadra Biswal [2] Jharana Paikaray [3]
Bikash Chandra Nayak [4]
[1,2,3,4] Dept. of CSE, Einstein Academy of Technology and Management, Bhubaneswar

Abstract:
Capability of traditional differential evolution algorithms for optimization solutions has been well documented and demonstrated in this paper. However, differential evolution is generally considered inappropriate for many Real world problems based on binary/permutation due to its arithmetic reproduction operator. Also the standard differential evolution algorithm has many limitations as slow convergence and local optima become trapped. In this paper, a novel technique that adapts a simple differential evolution algorithm. It is proposed to be very effective in solving binary-based problems, such as binary knapsacks. This includes new elements, such as the presentation of solutions, a mapping diversity method and technique. Furthermore, a new efficient fitness evaluation method for calculating and at the same time, to repair candidate solution of Knapsack is introduced.

Keywords: *Differential evolution (DE), Combinatorial Optimization Problem (COP)*

I. Introduction

Differential evolution (DE) is a member of the family of evolutionary algorithms (EAs). It is a stochastic population-based search technique originally developed by Storn and Price [1] to solve optimization problems in continuous domains using real numbers to represent the variables' values. It has a long history of successfully solving continuous optimization problems and is considered one of the best EAs for handling those with real-valued variables because of its simple structure, robustness, speed and ease of use. Due to its powerful features, DE has been applied to solve a wide range of optimization problems in different areas, such as clustering [2], power control systems [3] and chemical engineering [4] as well as for simultaneous transit network design [5] and several other practical applications as reviewed in [6]. It is used to optimize a problem by iteratively trying to improve a candidate solution over several generations through adaptation, emergence and learning (evolutionary) operations. Although, like most general EAs, DE is a population-based algorithm, it traditionally produces new solutions in every iteration by perturbing current candidate ones using a scaled difference between two other solutions randomly selected from the population whereas other EAs recombine solutions under conditions imposed by a probabilistic scheme. In DE, the concept of natural selection in biology is mimicked by allowing individuals with good fitness values to survive from generation to generation, with the quality of each solution evaluated using a pre-defined fitness function. A knapsack problem (KP) is a combinatorial optimization problem (COP) with binary decision variables which has proven to be an NP-complete problem [7]. In it, there is a set of items, with each item (x) having weight (w) and profit (p) values, and the main objective is to select those items that maximize the total profit without exceeding the knapsack's capacity. KPs arise in many practical applications in various fields, such as project selection [8], decision-making [9], water resource engineering and flood management [10], and the distribution and allocation of resources [11]. In a typical KP, the knapsack has a positive integer volume as its total weight capacity (W) in which N different items may be placed, each of which (j) has a positive integer weight (wj) and positive integer profit (pj).

Due to its practical value in numerous disciplines, the KP has been extensively studied as a discrete programming problem. In recent years, several exact and heuristic techniques for solving binary KPs (0–1 KPs) have been proposed. Initially, exact methods, such as linear programming (LP) [12], branch and bound (B&B) [13] and dynamic programming (DP) [14], were applied to address them. However, although they could achieve exact optimal solutions for many small KPs, they were considered inapplicable for large instances because the computational time they required increased exponentially with the problem's size. The most recent exact method for solving KPs was developed by Zenarosa et al. [15] who considered the bi-level quadratic KP (BQKP) as a test problem. On the other hand, several heuristic algorithms, which are viewed as advanced optimization methodologies for solving large-scale optimization problems with various complex situations, have been introduced in the literature. Evolutionary computation-based algorithms are the most popular examples of heuristic techniques, several versions of which for solving 0–1 KPs, such as genetic algorithms (GAs) [16], DE [17], particle swarm intelligence [18], the firefly algorithm (FA) [19], cuckoo search (CS) [20], ant colony

optimization (ACO) [21] and harmony search (HS) [22], have been proposed and, in recent years, new variants of heuristic algorithms introduced. In 2018, Feng et al. [23] proposed a novel chaotic monarch butterfly optimization (MBO) algorithm for solving 0–1 KPs in which chaos theory was applied in order to improve its global optimization capability. Another recent variant introduced by Feng et al. [24] applied a discrete version of the moth search algorithm (MSA) for solving KPs which demonstrated the importance of using a discretization function to improve the quality of solutions. Recent discrete versions of DE have been successfully applied to solve instances of COPs, such as traveling salesman problems (TSPs) [25], multi-skill resource-constrained project [26] and flow-shop [27] scheduling problems and showed very promising results. Also, versions of DE for solving different instances of complex resource-constrained project scheduling problems have achieved outstanding performances compared with those of other algorithms [28,29]. Although DE has presented an effective way of solving COPs and achieved (near-) optimal solutions within a reasonable time, further enhancements are still required to improve its performance in many aspects, such as its capability to optimally solve binary/discrete problems with large dimensions in a reasonable amount of time, speed up its convergence and avoid becoming trapped in local optima. Motivated by the defined limitations of the standard DE in the literature but DE's potential for solving other COPs, a new design adds components to it to enhance its performance and make it suitable for solving binary KPs. In order to cope with the continuous nature of DE processes, a mapping method, which uses the average continuous value of each vector, is applied to map every gene in that vector to its binary value immediately after its generation. Also, representations for maintaining both the continuous and binary values of each vector are proposed as any change in one side will affect the other, and a new way of repairing and calculating the fitness value of a KP's solution is introduced. In this method, the feasibility of each solution is checked and the solutions placed in feasible and infeasible groups, with those in the former improved and those in the latter (which violate one or more of a problem's constraints) repaired. Also, a diversity mechanism is applied to prevent the DE from becoming stuck in a local optimum solution and maintain the diversity of the population. Ultimately, the performances of this novel binary DE (NBin-DE) algorithm for solving 44 instances of KPs with different numbers of items are compared with those of 22 state-of-the-art algorithms.

II. Typical differential evolution (DE)

DE is considered a powerful tool for solving optimization problems in continuous spaces and has also proven to be a more robust and less biased optimization model than some other EAs, such as GAs and particle swarm optimization (PSO) [30]. It has three main parameters that need to be tuned, the scale factor (F), crossover rate (cr) and population size (PS). In the lastfew decades, several research efforts have attempted to realize the optimal values of these parameters and study their effectson the performance of DE [31]. Based on its original formulation provided by Storn and Price [1], DE randomly generates an initial PS of individuals, with a uniform distribution in the decision space (N). Then, it optimizes a problem by iteratively trying to improve each individual (x!i) in the population using three evolutionary operators, mutation, crossover and selection,which are applied in an iterative way to direct the search towards (near-) optimal solutions. The three major operators of thetypical version of DE are described in the following sub-sections.

III. Proposed binary DE algorithm

As a traditional DE was developed mainly to solve problems in continuous domains and so can handle only real-valued problems using its current operators, it is inapplicable for solving several permutation-based real-world combinatorial problems. Although some techniques using DE to solve discrete problems have been proposed, they are only suitable for small ones and need several enhancements to improve their performances. To overcome these issues, an efficient version of DE which incorporates useful and enhanced components, such as 1) an improved solution representation and mapping approach, 2) a new repairing and fitness calculation method and 3) a population diversity mechanism, is proposed in this paper.

Algorithm 2. Pseudo-code of proposed repairing and evaluation methods

```
Input: PS, bin_Pop, con_Pop, profits, weights, W, fit_eval
Output: bin_Pop, con_Pop, fit_eval, Fitness
Ratio ← Calculate the ratio values between the profits and weights of each item
for i = 1 to PS do
  vio_i = sum(bin_Pop_i × weights_i) − W
  if vio_i > 0 then
    while (vio_i > 0) do
      TakenItems_i = all bin_Pop_i with ones
      k ← Find the item with the lowest Ratio and included in TakenItems_i
      Put bin_Pop_i^k ← 0
      Put con_Pop_i^k ← 0
      vio_i = sum(bin_Pop_i × weights_i) − W
    end while
  else if vio_i < 0 then
    while (vio_i > 0) do
  UnTakenItems_i = select all bin_Pop_i with zeros and their weights_i ≤ −vio_i
  m ← Find the item with the largest Ratio and existed in UnTakenItems_i
    Put bin_Pop_i^m ← 1
    Put con_Pop_i^m ← 0
  end while
else (vio_i = 0)
    Find all items with the same weights
    ItemsSameWeight ←Rank them, in descending order, according to their profits
    for each item in ItemsSameWeight with bin_Pop_i = 1 do
      if the Ratio of this item is less than any other items in ItemsSameWeight with bin_Pop_i = 0 then
        Replace the two items
      end if
    end for
  end if
  vio_i = sum(bin_Pop_i × weights_i) − W
  Total_profit_i = sum(bin_Pop_i × profits)
  Fitness_i=Total_profit_i − max(vio_i, 0)
  fit_eval = fit_eval + 1
end for
```

Fig 1. Pseudo code for proposed model

IV. Experimental results

To demonstrate the effectiveness of our proposed NBin-DE algorithm, which was coded in MATLAB R2017b and run on a PC (i7 processor and 16 GB memory), several benchmark binary KPs (0-1KPs) with different dimensions were used. They were classified in four groups based on their sizes and sources, with the previously developed techniques used to solve them.

For each KP, the algorithm was executed 30 times with 5,000 fitness evaluations in each run. In order to assess its performance and those of other algorithms, AE%, which is the average error between the solution obtained by an algorithm and the optimal/best-known one for that problem, was used for comparison and calculated. The proposed NBin-DE significantly outperformed the average of the other algorithms in terms of total profits when solving

f1, f2, f5 and f10 and, although there was no significant difference for f7, it could solve it to optimality.

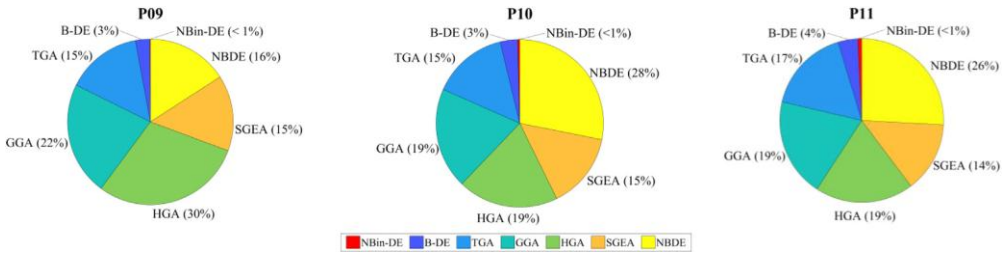

Fig. 2. Pie charts showing computational times for KP09, KP10 and KP11 taken by NBin-DE and six other algorithms

References

[1]. R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359.

[2]. Y. Zhou, Z. Zhao, D. Cheng, Cluster structure prediction via revised particle-swarm optimization algorithm, Comput. Phys. Commun. 247 (2020)106945.

[3]. S. Panda, "Joint User Patterning and Power Control Optimization of MIMO–NOMA Systems," Wireless Personal Communications, pp. 1-17, 2020.

[4]. E.N. Dragoi, S. Curteanu, The use of differential evolution algorithm for solving chemical engineering problems, Rev. Chem. Eng. 32 (2016) 149–180.
[5]. A.T. Buba, L.S. Lee, A differential evolution for simultaneous transit network design and frequency setting problem, Expert Syst. Appl. 106 (2018) 277–289.
[6]. V. Plagianakos, D. Tasoulis, and M. Vrahatis, "A review of major application areas of differential evolution," in Advances in differential evolution, ed:Springer, 2008, pp. 197-238.
[7]. D.-Z. Du, K.-I. Ko, X. Hu, Design and analysis of approximation algorithms, vol. 62, Springer Science & Business Media, 2011.
[8]. G. Mavrotas, D. Diakoulaki, A. Kourentzis, Selection among ranked projects under segmentation, policy and logical constraints, Eur. J. Oper. Res. 187(2008) 177–192.
[9]. S. Peeta, F.S. Salman, D. Gunnec, K. Viswanath, Pre-disaster investment decisions for strengthening a highway network, Comput. Oper. Res. 37 (2010)1708–1719.
[10]. Z.M. Yaseen, S.O. Sulaiman, R.C. Deo, K.-W. Chau, An enhanced extreme learning machine model for river flow forecasting: State-of-the-art, practicalapplications in water resource engineering area and future research direction, J. Hydrol. 569 (2019) 387–408.
[11]. D.C. Vanderster, N.J. Dimopoulos, R. Parra-Hernandez, R.J. Sobie, Resource allocation on computational grids using a utility model and the knapsackproblem, Future Gen. Computer Systems 25 (2009) 35–50.
[12]. S. Senju, Y. Toyoda, An approach to linear programming with 0–1 variables, Manage. Sci. (1968) B196–B207.
[13]. W. Shih, A branch and bound method for the multiconstraint zero-one knapsack problem, J. Operat. Res. Soc. (1979) 369–378.
[14]. P. Toth, Dynamic programming algorithms for the zero-one knapsack problem, Computing 25 (1980) 29–45.
[15]. G.L. Zenarosa, O.A. Prokopyev, E.L. Pasiliao, On exact solution approaches for bilevel quadratic 0–1 knapsack problem, Ann. Oper. Res. (2018).
[16]. L. Soukaina, N. Mohamed, E.A. Hassan, A. Boujemâa, A hybrid genetic algorithm for solving 0/1 Knapsack Problem, in: Proceedings of the International Conference on Learning and Optimization Algorithms: Theory and Applications, 2018, p. 51.
[17]. I.M. Ali, D. Essam, K. Kasmarik, An efficient differential evolution algorithm for solving 0–1 Knapsack Problems, IEEE Cong. Evolution. Comput. (CEC)2018 (2018) 1–8.
[18]. C.-J. Lin, M.-S. Chern, M. Chih, A binary particle swarm optimization based on the surrogate information with proportional acceleration coefficients forthe 0–1 multidimensional knapsack problem, J. Indust. Product. Eng. 33 (2016) 77–102.
[19]. Y. Feng, G.-G. Wang, L. Wang, Solving randomized time-varying knapsack problems by a novel global firefly algorithm, Eng. Computers 34 (2018) 621–635.
[20]. Y. Feng, G.-G. Wang, X.-Z. Gao, A novel hybrid cuckoo search algorithm with global harmony search for 0–1 knapsack problems, Int. J. Comput. Intell.Systems 9 (2016) 1174–1190.
[21]. W. Zouari, I. Alaya, M. Tagina, A hybrid ant colony algorithm with a local search for the strongly correlated knapsack problem, in: Proceeding ComputerSystems and Applications (AICCSA), 2017 IEEE/ACS 14th International Conference on, 2017, pp. 527–533.
[22]. M. El-Shafei, I. Ahmad, M.G. Alfailakawi, Hardware accelerator for solving 0–1 knapsack problems using binary harmony search, Int. J. ParallelEmergent Distrib. Syst. 33 (2018) 87–102.
[23]. Y. Feng, J. Yang, C. Wu, M. Lu, X.-J. Zhao, Solving 0–1 knapsack problems by chaotic monarch butterfly optimization algorithm with Gaussian mutation,Memetic Comput. 10 (2018) 135–150.
[24]. Y. Feng, H. An, X. Gao, The importance of transfer function in solving set-union knapsack problem based on discrete moth search algorithm,Mathematics 7 (2019) 17.
[25]. I.M. Ali, D. Essam, K. Kasmarik, A novel design of differential evolution for solving discrete traveling salesman problems, Swarm Evol. Comput. 52(2020) 100607.
[26]. P.B. Myszkowski, Ł.P. Olech, M. Laszczyk, M.E. Skowron´ ski, Hybrid differential evolution and greedy algorithm (DEGR) for solving multi-skill resourceconstrainedproject scheduling problem, Appl. Soft Comput. 62 (2018) 1–14.
[27]. B.-H. Zhou, L.-M. Hu, Z.-Y. Zhong, A hybrid differential evolution algorithm with estimation of distribution algorithm for reentrant hybrid flow shopscheduling problem, Neural Comput. Appl. 30 (2018) 193–209.
[28]. I.M. Ali, S.M. Elsayed, T. Ray, R.A. Sarker, Memetic algorithm for solving resource constrained project scheduling problems, Proceeding IEEE Congress onEvolutionary Computation (CEC) 2015 (2015) 2761–2767.
[29]. H.F. Rahman, R.K. Chakrabortty, M.J. Ryan, Memetic algorithm for solving resource constrained project scheduling problems, Autom. Constr. 111(2020) 103052.
[30]. F. Caraffini, A.V. Kononova, D. Corne, Infeasibility and structural bias in differential evolution, Information Sci. 496 (2019) 161–179.
[31]. A. Yaman, G. Iacca, F. Caraffini, "A comparison of three differential evolution strategies in terms of early convergence with different population sizes",in proceeding, AIP Conf. Proc. (2019) 020002.
[32]. A. Qing, Dynamic differential evolution strategy and applications in electromagnetic inverse scattering problems, IEEE Trans. Geosci. Remote Sens. 44(2005) 116–125.
[33]. I.M. Ali, D. Essam, K. Kasmarik, A novel differential evolution mapping technique for generic combinatorial optimization problems, Appl. Soft Comput.80 (2019) 297–309.
[34]. J. Liang, W. Xu, C. Yue, K. Yu, H. Song, O.D. Crisalle, B. Qu, Multimodal multiobjective optimization with differential evolution, Swarm Evol. Comput. 44(2019) 1028–1059.
[35]. A.W. Mohamed, A new modified binary differential evolution algorithm and its applications, Appl. Math. Inform. Sci. 10 (2016) 1965–1969.
[36]. K.K. Bhattacharjee, S.P. Sarmah, Shuffled frog leaping algorithm and its application to 0/1 knapsack problem, Appl. Soft Comput. 19 (2014) 252–263.
[37]. N. Abd-Alsabour, Investigating the influence of adding local search to search algorithms, in: Proceeding Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2017 18th International Conference on, 2017, pp. 145-150.
[38]. Wikipedia. (2019, Broyden-Fletcher-Goldfarb-Shanno algorithm - Wikipedia. Available: https://en.wikipedia.org/wiki/Broyden%E2%80%93Fletcher%E2%80%93Goldfarb%E2%80%93Shanno_algorithm.
[39]. L. Zhang, B. Zhang, Good point set based genetic algorithm, Chin. J. Computers 24 (2001) 917–922.
[40]. Y. He, K. Liu, C. Zhang, W. Zhang, Greedy genetic algorithm for solving knapsack problems and its applications, Computer Eng. Design 28 (2007) 2655–2657.
[41]. S. Jun and L. Jian, "Solving 0-1 knapsack problems via a hybrid differential evolution," in proceeding Intelligent Information Technology Application,2009. IITA 2009. Third International Symposium on, 2009, pp. 134-137.
[42]. Y. Feng, G.-G. Wang, S. Deb, M. Lu, X.-J. Zhao, Solving 0–1 knapsack problem by a novel binary monarch butterfly optimization, Neural Comput. Appl.28 (2017) 1619–1634.

[43]. Y. Chen, W. Xie, X. Zou, A binary differential evolution algorithm learning from explored solutions, Neurocomputing 149 (2015) 1038–1047.
[44]. T. Gong, A.L. Tuson, Differential evolution for binary encoding, in: Soft Computing In Industrial Applications, Springer, 2007, pp. 251–262.
[45]. A. R. Hota and A. Pat, ‘‘An adaptive quantum-inspired differential evolution algorithm for 0–1 knapsack problem,’’ in Proceeding Nature andBiologically Inspired Computing (NaBIC), 2010 Second World Congress on, 2010, pp. 703-708.
[46]. H. Peng, Z. Wu, P. Shao, C. Deng, Dichotomous binary differential evolution for knapsack problems, Math. Problems Eng. 2016 (2016).
[47]. J. Kennedy and R. C. Eberhart, ‘‘A discrete binary version of the particle swarm algorithm,’’ in proceeding Systems, Man, and Cybernetics, 1997.Computational Cybernetics and Simulation., 1997 IEEE International Conference on, 1997, pp. 4104-4108.
[48]. J.C. Bansal, K. Deep, A modified binary particle swarm optimization for knapsack problems, Appl. Math. Comput. 218 (2012) 11042–11061.
[49]. R. Woolson, ‘‘Wilcoxon signed-rank test,’’ Wiley encyclopedia of clinical trials, pp. 1-3, 2007.
[50]. M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, Ann