Discrete differential evolutions for the discounted {0–1} knapsack problem

Hong Zhu

Faculty of Information Technology,
Macau University of Science and Technology,
Macau, China
Email: xszhuhong@163.com

Yichao He

College of Information and Engineering,
Hebei GEO University,
050031, Shijiazhuang, China
Email: heyichao@hgu.edu.cn

Xizhao Wang*

College of Computer Science and Software Engineering,
Shenzhen University,
518060, Shenzhen, China
Email: xizhaowang@ieee.org
*Corresponding author

Eric C.C. Tsang

Faculty of Information Technology,
Macao University of Science and Technology,
Macau, China
Email: cctsang@must.edu.mo

Abstract: This paper first proposes a discrete differential evolution algorithm for discounted {0–1} knapsack problems (D{0–1}KP) based on feasible solutions represented by the 0–1 vector. Subsequently based on two encoding mechanisms of transforming a real vector into an integer vector, two new algorithms for solving D{0–1}KP are given through using integer vectors defined on {0, 1, 2, 3}n to represent feasible solutions of the problem. Finally the paper conducts a comparative study on the performance between our proposed three discrete differential evolution algorithms and those developed by common genetic algorithms for solving several types of large scale D{0–1}KP problems. The paper confirms the feasibility and effectiveness of designing discrete differential evolution algorithms for D{0–1}KP by encoding conversion approaches.

Keywords: discounted {0–1} knapsack problem; differential evolution; encoding conversion method; repairing and optimisation.

Reference to this paper should be made as follows: Zhu, H., He, Y., Wang, X. and Tang, E.C.C. (xxxx) 'Discrete differential evolutions for the discounted {0–1} knapsack problem', ***Int. J.*** *Bio-Inspired Computation*, Vol. X, No. Y, pp.xxx–xxx.

Biographical notes: Hong Zhu received her BSc and MSc degrees from Hebei University, Baoding, China, in 2012 and 2015, respectively. She is currently a PhD candidate at the Faculty of Information Technology, Macao University of Science and Technology. Her main research interests include decision tree and neural networks.

Yichao He is a Professor at Hebei GEO University, China. He is a CCF member and CAAI member. His research interest includes evolutionary algorithm and its applications, approximation algorithm, combinatorial optimization, group testing theory. He has published over 60 publications including *Information Sciences*, *Journal of Combinatorial Optimization*, *Journal of Computational Biology*, *Chinese Journal of Computers* and so on.

Copyright © 20XX Inderscience Enterprises Ltd.

Xizhao Wang received his PhD in Computer Science from the Harbin Institute of Technology on September 1998. His major research interests include uncertainty modelling and machine learning for big data. He has edited 10+ special issues and published three monographs, two textbooks, and 200+ peer-reviewed research papers. By the Google Scholar, the total number of citations is over 5,000 and the maximum number of citation for a single paper is over 200. He is on the list of Elsevier 2015/2016 most cited Chinese authors. As a Principle Investigator (PI) or Co-PI, he has completed 30+ research projects.

Eric C.C. Tsang received his BSc degree in Computer Studies from the City University of Hong Kong in 1990 and PhD degree in Computing at the Hong Kong Polytechnic University in 1996. He is an Associate Professor of the Faculty of Information Technology, Macao University of Science and Technology. His main research interests include fuzzy systems, rough sets, fuzzy rough sets and multiple classifier systems.

1 Introduction

Differential evolution (DE), proposed by Storn and Price (1997a, 1997b, 2005), is a powerful evolutionary algorithm (EA) for optimisation problems (Ekbal and Saha, 2016; Abe, 2016; Souravlias and parsopoulos, 2016). It not only has the general characteristics of EAs (Bäck et al., 2000; Yao et al., 1999), such as robust and reliable performance, global search capability and little or no information requirement of optimisation problem, etc., but also has less control parameters and is easy to be implemented. At present, people have done a lot of researches on DE. For example, Qin et al. (2009) did the adaptive adjustment of the control parameters *F* and *CR* based on the optimal individual obtained in the previous work of DE and then proposed an improved DE named SaDE. Kaelo and Ali (2006) used the tournament competitive mechanism to generate a new population and improved the local search ability by the reflection and shrink operations. Noman and Iba (2008) introduced the adaptive local search operation to improve the convergence speed of DE. Das et al. (2009) introduced the neighbourhood mutation to enhance the local search ability of DE. Wang et al. (2011) improved the global and local search capabilities based on trial vector generation strategies and control parameters. Zhang et al. (2013) improved the local search capability of DE based on abstract convex lower approximation. Fan and Lampinen (2003) introduced trigonometry mutation, which can improve the probability of DE jumping out of local extreme points. He et al. (2010) analysed the asymptotic convergence of DE and divided the mutation strategy into three equivalence classes to achieve cooperative operations, which can improve the global optimisation ability of DE. Based on Pareto competition, Abbas et al. (2001) and Abbas (2002) proposed self-adaptive Pareto DE algorithm to solve multi-objective optimisation problems. Simulation results showed that the algorithm could get better Pareto solutions. In order to deal with integer programming problems by means of DE, Nearchou and Omirou (2006) proposed a method to solve the sequence and schedule problem by DE which uses the sub-range encoding method. He and Han (2007) used 0–1 string to represent individuals; they replaced the arithmetic operations in the standard DE with logic operations and proposed a binary DE. Based on the encoding conversion method which can transfer the real vector into a binary vector, He et al. (2007) proposed a hybrid-encoding binary differential evolution algorithm (HBDE) to solve the 0–1 knapsack problem (0–1KP) and satisfiability problem (SAT). Greenwood (2009) converted a real vector into a binary vector based on the method of transferring real number segmentation into binary number and proposed a binary DE and used it to solve problems in graph theory. Due to the good searching ability, DE has been widely used to solve many combinatorial optimisation problems.

Knapsack problem (KP) (Kellerer et al. 2004; Du and Ko, 2000) is an important combinatorial optimisation problem (Sarkar et al. 2016; De et al., 2015; Tian et al., 2015) and it is also a classic NP-complete problem in computer science. It has an important application background in investment decision-making and resource allocation (Azada et al., 2014; Haddar et al., 2016). There are many different classical extended forms of KP, such as bounded knapsack problem (BKP), unbounded knapsack problem (UKP), multidimensional knapsack problem (MKP), multiple-choice knapsack problem (MCKP), quadratic knapsack problem (QKP) (Kellerer et al., 2004) and 0–1 KP (Kulkarni, 2016), etc. Because KP is an NP-complete problem, exact algorithms with polynomial time complexity does not exist unless P = NP. Therefore, the non-exact algorithms with polynomial time complexity are paid more attention to. Currently, numerous studies have shown that EAs is a class of stochastic approximation algorithms that are suitable for solving combinatorial optimisations and it has been successfully used to solve KP. For example, He et al. (2001) used HBDE to solve 0–1 KP; Lai et al. (2014) solved MKP by using genetic algorithms (GA) (Zhang et al., 2015; Martínez-Soto and Castillo, 2015); Kong et al. (2008) proposed the ant colony optimisation algorithm to solve MKP; Azad et al. (2014) solved the 0–1 QKP by a binary artificial fish swarm algorithm; Chih et al. (2014) advanced a particle swarm optimisation with time-varying acceleration coefficients to solve MKP. Therefore, EAs is an obviously effective method for solving KP.

Recently, many expanded forms of KP have been proposed one after another, such as stochastic knapsack problem (SKP), dynamic knapsack problem (DKP), 0–1KP with a single continuous variable (KPC) and discounted {0–1} knapsack problem (D{0–1}KP) (Lin et al., 2008; Dizdar et al., 2011; Goldberg and Smith, 1987; Haddad and Erick, 1997; He et al., 2016, 2017; Marchand and Wolsey, 1999; Lin et al., 2011; Zhao and Li, 2014; Guldan, 2007; Rong et al., 2012), which have more practical backgrounds and begin to attract people's attention. For example, Lin et al. (2008) studied deeply the exchange policy and dynamic pricing problem of SKP; Dizdar et al. (2011) researched the applications of SKP to the tax maximisation; Goldberg and Smith (1987) proposed the time-varying knapsack problems (TVKP) in which the capacity of knapsack oscillates between two fixed values and they solved TVKP by the use of GA (Sun and Shen, 2016) with diploid form; Hadad and Lewis et al. (1997) solved TVKP by using the GA which has a polyploid form individually and compared the advantages and disadvantages of several polyploid forms of an individual; He et al. (2016, 2017) extended TVKP to randomised time-varying knapsack problems (RTVKP) and solved RTVKP by using dynamic programming, approximation algorithm and GA separately; Marchand and Wolsey (1999) proposed KPC and analysed its mathematical properties; Lin et al. (2011) gave a deterministic algorithm for solving KPC; Zhao and Li (2014) proposed an 2-approximation algorithm for solving KPC; Guldan (2007) proposed D{0–1}KP and gave a dynamic programming algorithm to solve it; Rong et al. (2012) studied the core problem of D{0–1}KP and combined it with the dynamic programming to solve D{0–1}KP; He et al. (2016) proposed a new mathematical model for D{0–1}KP and two effective algorithms, named FirEGA and SecEGA, by using GA and they indicated the performance of FirEGA is better than ones of SecEGA.

The D{0–1}KP (Guldan, 2007) is a variant of the classical 0–1KP by extending the number of choices for each item group based on the concept of discount. The discount discussed here originates from economies of scale, which refers to the cost advantages that a business obtains due to expansion. Although the D{0–1}KP has not received much attention in the literature, the discount introduced in the D{0–1}KP is close to the reality of the real world problem. Economies of scale are a practical concept that may explain real world phenomena such as patterns of international trade and the investment scales of the business (Rong et al., 2012). It means that the D{0–1}KP may find applications in investment, project selection and budget control. The number of choices for the D{0–1}KP in each item group is four: either one of the three items is selected or no item is selected. On the one hand, if an item group is selected, it needs to be determined which item in the group is selected. On the other hand, the condition for not selecting an item group is hardly known since the weight and profit range of the three items may be large. It implies that D{0–1}KP is harder than 0–1 KP. In addition, because the exact algorithms (Guldan, 2007; Rong et al., 2012) solving D{0–1}KP have all pseudo-polynomial time complexity, for a large number of D{0–1}KP instances with profit and weight coefficients distributing in larger intervals, the high time complexity leads to poor usability of this algorithms. As a matter of fact, the NP-hard problems in practical application are almost always required to be solved fast. The exact solution is not necessary and only one approximate solution is needed which satisfies the approximate ratio requirement (Michael, 2002). Noting many successful applications of EAs to solving 0–1KP, we believe that using EAs to solve D{0–1}KP is an inexpensive and efficient method which is worth being explored. For solving D{0–1}KP by HBDE is given based on our previous work (He et al., 2007) firstly; then, two discrete DE algorithms, named FDDE and SDDE, are proposed by using the encoding conversion method.

The rest of the paper is organised as follows: in Section 2 the definition and mathematical models of D{0–1}KP is introduced. In Section 3, based on the first mathematical model of D{0–1}KP, the binary DE algorithm HBDE is given by combining with GROA (He et al., 2016) (see Appendix 1). In Section 4, based on the second mathematical model of D{0–1}KP, the FDDE and SDDE are introduced separately by using the encoding conversion method and the NROA algorithm (He et al., 2016) (see Appendix 2). In Section 5, four types of large-scale D{0–1}KP instances discussed in (He et al., 2016) are calculated with HBDE, FDDE and SDDE. Then the result is compared with those of FirEGA and SecEGA (He et al., 2016). Based on the comparison and analysis of the results, we indicate that HBDE, FDDE and SDDE are more suitable for solving all kinds of D{0–1}KP instances than FirEGA and SecEGA. Moreover, it is not only feasible, but also efficient to discretise the DE based on the coding transformation. At last, the whole content of the paper is summarised and further research ideas are prospected.

2 Definition and mathematical models of D{0–1}KP

Guldan (2007) proposed D{0–1}KP in 2007 and established its first mathematical model based on the linear programming theory; He et al. (2016) put forward the second and the third mathematical model of D{0–1}KP based on the integer programming theory. Since in this paper, we will discuss how to use DE to solve D{0–1}KP based on the first and the second mathematical model of D{0–1}KP and the definition of D{0–1}KP will be proposed first and then its two mathematical models will be introduced.

Definition: D{0–1}KP (He et al., 2016):

Given a set of n item groups and each group i ($i = 0, 1, \ldots, n - 1$) consists of three items $3i$, $3i + 1$ and $3i + 2$ and the first two items $3i$ and $3i + 1$ with weights w_{3i} and w_{3i+1} and profits p_{3i} and p_{3i+1} are paired to derive a third item $3i + 2$ with discounted weight $w_{3i+2} < w_{3i} + w_{3i+1}$ and profit $p_{3i+2} = p_{3i} + p_{3i+1}$. In each group, at most one of the three items can be selected to be placed in the knapsack with capacity C so that the total weight of the selected items cannot exceed C and the total profit is maximised.

Suppose the scale of D{0–1}KP instances is the number of the items, $3n$. The D{0–1}KP instant consists of profit coefficient set $P = \{p_j | 0 \le j \le 3n - 1\}$, weight coefficient set $W = \{w_j | 0 \le j \le 3n - 1\}$ and the knapsack capacity C. Without loss of generality, it may be assumed that all profit coefficients (p_{3i}, p_{3i+1} and p_{3i+2}), weight coefficients (w_{3i}, w_{3i+1} and w_{3i+2}) and knapsack capacity C are positive integers and all the weight coefficients are not larger than the capacity C, $\sum_{i=0}^{n-1} w_{3i+2} > C$.

2.1 First mathematical model

Let $X = [x_0, x_1, \ldots, x_{3n-1}] \in \{0, 1\}3n$ be a binary vector. The first mathematical model of D{0–1} (Guldan, 2007; Rong et al., 2012) KP is:

$$\text{Maximize} \quad f(X) = \sum_{i=0}^{n-1} \begin{pmatrix} x_{3i} p_{3i} + x_{3i+1} p_{3i+1} \\ + x_{3i+2} p_{3i+2} \end{pmatrix} \tag{1}$$

$$\text{Subject to} \quad x_{3i} + x_{3i+1} + x_{3i+2} \le 1, \quad i = 0, 1, \ldots, n-1 \tag{2}$$

$$\sum_{i=0}^{n-1} \left(x_{3i} w_{3i} + x_{3i+1} w_{3i+1} + x_{3i+2} w_{3i+2} \right) \le C \tag{3}$$

$$x_{3i}, x_{3i+1}, x_{3i+2} \in \{0, 1\}, \quad i = 0, 1, \ldots, n-1. \tag{4}$$

Where, the binary decision variables $x_j (0 \le j \le 3n - 1)$ are used to indicate whether the item j is included in the knapsack or not. The item j is loaded into the knapsack if and only if $x_j = 1$. Obviously, any 0–1 vector $X = [x_0, x_1, \ldots, x_{3n-1}] \in \{0, 1\}3n$ merely represents a potential solution to D{0–1}KP. It is a feasible solution only when it satisfies the constraints (2) and (3) at the same time. Otherwise, it is an infeasible solution to D{0–1}KP.

2.2 Second mathematical model

Let $X = [x_0, x_1, \ldots, x_{n-1}] \in \{0, 1, 2, 3\}^n$ be an integer vector. The second mathematical model of D{0–1}KP (He et al., 2016) is:

$$\text{Maximize} \quad f(X) = \sum_{i=0}^{n-1} \lceil x_i / 3 \rceil \, p_{|3i + (x_i - 1)|} \tag{5}$$

$$\text{Subject to} \quad \sum_{i=0}^{n-1} \lceil x_i / 3 \rceil \, w_{|3i + (x_i - 1)|} \le C \tag{6}$$

$$x_i \in \{0, 1, 2, 3\}, \quad i = 0, 1, \ldots, n-1. \tag{7}$$

Where, $\lceil x \rceil$ is a top function. The integer variables $x_i (0 \le i \le n - 1)$ indicate whether there is an item of the item group i to be loaded into the knapsack or not. No items of item group i is loaded into the knapsack when $x_i = 0$. The item $3i$ is loaded into the knapsack when $x_i = 1$. The item $3i + 1$ is loaded into the knapsack when $x_i = 2$. The item $3i + 2$ is loaded into the knapsack when $x_i = 3$. Obviously, arbitrary integer vector $X = [x_0, x_1, \ldots, x_{n-1}] \in \{0, 1, 2, 3\}^n$ only represents a potential solution of D{0–1}KP and it is a feasible solution to the problem if and only if it satisfies the inequality (6).

3 Binary DEs for solving D{0–1}KP

In the first mathematical model of D{0–1}KP, the feasible solution is a binary vector, but the individual coding of the standard DE is a real vector. So it is impossible to solve D{0–1}KP by using the standard DE directly. Therefore, a binary version DE named HBDE for solving D{0–1}KP is proposed based on our previous work (He et al., 2007) and we will use the algorithm GROA introduced in (He et al., 2016) to handle infeasible solutions of D{0–1}KP in HBDE.

To solve the D{0–1}KP, we make four improvements for HBDE as follows:

1. Firstly, we use a $3n$-dimensional real vector to represent an individual in HBDE. To get a potential solution of D{0–1}KP, we use the encoding conversion function to transform the $3n$-dimensional real vector into a 3n-dimensional binary vector.
2. Secondly, we give a simple implement method of encoding conversion function which equals to one in (He et al., 2007) and is easier to be implemented. The computational complexity can be greatly reduced when a real vector was converted into a binary vector.
3. By Gauss-Seidel method (Michael, 2002), the temporary population is no longer used in HBDE. We immediately compare the offspring individual with the parent individual after the offspring individual is generated through the mutation and crossover of standard DE. If the offspring individual is better than the parent one, then it replaces the parent individual immediately, otherwise remain the parent individual without change. This not only can make more new outstanding individuals participate in the evolution process as soon as possible but also decreases the space complexity of HBDE.
4. To make the potential solution (i.e., $3n$-dimensional binary vector) be a high-quality feasible solution, we use the GROA (He et al., 2016) algorithm to repair and optimise all individuals in HBDE. At the same time, the objective function value is calculated as the fitness of the corresponding individual.

The description of the algorithm principle and pseudo-code based on the DE/rand/1/bin model is shown as follows:

Let $X_i = [x_{i0}, x_{i1}, \ldots, x_{i,3n-1}] \in \mathbf{S}_1$ represents the i^{th} individual of the current population in HBDE, where $S_1 = \prod_{j=1}^{3n} [low, high]$, *low* and *high* are all real numbers, $low < 0$ and $low = -high$; $1 \le i \le N$, N is the population size; n is the number of item groups in the D{0–1}KP instance.

The encoding conversion function of HBDE is defined as $W = g_1(V)$. Its expression is described as follows:

$$w_j = \begin{cases} 1, & \text{if} \quad v_j \ge 0; \\ 0, & otherwise \end{cases} \tag{8}$$

where $V = [v_0, v_1, \ldots, v_{3n-1}] \in \mathbf{S}_1$, $W = [w_0, w_1, \ldots, w_{3n-1}] \in \{0, 1\}3n$, $v_j \in V$ and $w_j \in W$, $j = 0, 1, \ldots, 3n - 1$.

Let $Y = [y_0, y_1, \ldots, y_{3n-1}] \in \{0, 1\}3n$ represent the binary vector, which is obtained by using the encoding conversion function $Y = g_1(X)$, where X is an individual in HBDE. It is clear that Y is a potential solution of D{0–1}KP. We can use GROA to repair and optimise Y to make it be a feasible solution to D{0–1}KP and calculate $f(Y)$ as the fitness of individual X.

For example, suppose the scale of the D{0–1}KP instance $\mathbf{I}_1$ is $3n = 12$; $S_1 = \prod_{j=1}^{12}[-5.0, 5.0]$ and $X = [1.15, -4.73, 3.44, -2.32, -0.71, -1.08, 2.29, 4.11, -3.69, 3.15, -2.66, -4.01] \in \mathbf{S}_1$ is an individual in HBDE. The potential solution $Y = [1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0] \in \{0, 1\}12$ corresponding to X can be obtained by using $Y = g_1(X)$. The schematic diagram of $Y = g_1(X)$ is shown as follows:

For the i^{th} ($i = 1, 2, \ldots, N$) individual $X_i = [x_{i0}, x_{i1}, \ldots, x_{i,3n-1}] \in \mathbf{S}_1$ in the current population of HBDE, let $Z = [z_0, z_1, \ldots, z_{3n-1}] \in \mathbf{S}_1$ represent a temporary $3n$-dimensional real vector, which is used to derive the offspring individual of Xi in the following operation. Thereupon, the mutation and crossover operations of HBDE are achieved separately by using (9) and (10).

$$z_j = x_{p1,j} + F * \left(x_{p2,j} - x_{p3,j}\right) \tag{9}$$

$$z_j = \begin{cases} z_j, & \text{if } r < CR \text{ or } j = R(i); \\ x_{ij}, & \textit{otherwise.} \end{cases} \tag{10}$$

where $j = 0, 1, \ldots, 3n - 1$; X_{p1}, X_{p2} and Xp3 are three different individuals in the current population that have difference with X_i; scaling factor $F \in (0, 1]$, $r \sim (0, 1)$ is a random number; $R(i)$ represents a random positive integer in interval $[1, n]$; CR is called cross factor and $CR \in (0, 1)$.

In the selection operation of HBDE, we first use the encoding conversion function $U = g_1(Z)$ to transform the real vector $Z = [z_0, z_1, \ldots, z_{3n-1}]$ into a binary vector $U = [u_0, u_1, \ldots, u_{3n-1}] \in \{0, 1\}^{3n}$. Since U may not be a feasible solution to D{0–1}KP, we use GROA (He et al., 2016) to repair and optimise U, making it be a high-quality feasible solution. Then, we calculate the objective function value $f(U)$ as the fitness of Z and then use equation (11) to select between X_i and Z.

$$X_i = \begin{cases} Z, & \textit{if } f(U) > f(Y_i); \\ X_i, & \textit{otherwise.} \end{cases} \tag{11}$$

where, $Y_i = g_1(X_i)$ is a binary vector corresponding to X_i.

By the above description, the algorithm principle of HBDE is shown as follows:

1 Initialisation: generate the population $\mathbf{P}(0) = \{X_i(0) \in \mathbf{S}_1 \mid 1 \le i \le N)\}$ randomly. Use $Y_i(0) = g_1(X_i(0))$ to calculate the potential solution $Y_i(0) \in \{0, 1\}3n$. Repair and optimise $Y_i(0)$ $(1 \le i \le N)$ by using GROA. Then based on $f(Y_i(0))$ $(1 \le i \le N)$, determine the global optimal individual $X_b(0) = [x_{b0}(0), x_{b1}(0), \ldots, x_{b,3n-1}(0)] \in \mathbf{S}_1$ in $\mathbf{P}(0)$ and its corresponding feasible solution $Y_b(0) = [y_{b0}(0), y_{b1}(0), \ldots, y_{b,3n-1}(0)] \in \{0, 1\}^{3n}$. Let t be the iteration control variable and set $t = 0$.

2 The $(t + 1)^{\text{th}}$ iteration evolution process of HBDE: for each individual $X_i(t)$ $(1 \le i \le N)$ in $\mathbf{P}(t)$, we first generate a temporary $3n$-dimensional real vector $Z \in \mathbf{S}_1$ based on (9) and (10). Then, use $U = g_1(Z)$ to get the potential solution $U \in \{0, 1\}^{3n}$ corresponding to Z. We repair and optimise U by using GROA and calculate $f(U)$. If $f(U) > f(Y_i(t))$, we replace $(X_i(t), Y_i(t))$ with (Z, U); otherwise keep $(X_i(t), Y_i(t))$ unchanged. The new population $\mathbf{P}(t + 1)$ is generated when all the individuals of $\mathbf{P}(t)$ have finished the above operations. In $\mathbf{P}(t + 1) \cup \{(X_b(t), Y_b(t))\}$, determine the global optimal individual $X_b(t + 1) = [x_{b0}(t + 1), x_{b1}(t + 1), \ldots, x_{b,3n-1}(t + 1)] \in \mathbf{S}_1$ and its corresponding feasible solution $Y_b(t + 1) = [y_{b0}(t + 1), y_{b1}(t + 1), \ldots, y_{b,3n-1}(t + 1)] \in \{0, 1\}^{3n}$. Let $t = t + 1$.

3 Termination determination: If $t \le MaxIt$ ($MaxIt$ is the number of iterations of HBDE), go back to (2) to carry out the next iterative evolution; otherwise output $(Y_b(t - 1), f(Y_b(t - 1)))$ and end the algorithm.

Let 'H[0 … 3n – 1] ← ***Sort***$(\{p_j / w_j \mid p_j \in P, w_j \in W, 0 \le j \le 3n - 1\})$' represent the procedure that sequentially store the original index of each item into the array H[0 … 3n – 1] after $3n$ items are sorted according to the p_j / w_j $(0 \le j \le 3n - 1)$ descending order. Let $rand(0, 1)$ be a random number in (0, 1). The pseudo-code of HBDE is described as follows:

Algorithm 1 HBDE

Input: The D{0–1}KP instances, parameters N, $MaxIt$, low, $high$, F and CR;

Output: Approximate (or optimal) solution $Y_b(t - 1)$ and its objective function value $f(Y_b(t - 1))$.

```
H[0 … 3n – 1] ← Sort({p_j / w_j | p_j ∈ P, w_j ∈ W, 0 ≤ j ≤ 3n – 1});
Generate initial population P(0)={X_i(0) ∈ S_1 | 1 ≤ i ≤ N} randomly;
    for i = 1 to N do
        Y_i(0) ← g_1(Xi(0));
        (Y_i(0), f(Y_i(0))) ← GROA(Y_i(0), H[0 … 3n – 1])
    end for
    Determine (X_b(0), Y_b(0)) by f(Y_i(0)) (1 ≤ i ≤ N) in P(0); t ← 0;
    while (t < MaxIt) do
        for i = 1 to N do
            for j = 0 to 3n – 1do
                if (r ≤ CR∨j = R(i)) then z_j ← y_p1,j(t) + F*(y_p2,j(t) – y_p3,j(t)) else z_j ← y_ij(t);
                if (z_j < low or z_j > high) then z_j ← rand(0, 1) * (high – low) + low;
                if z_j ≥ 0 then u_j ← 1 else u_j ← 0;
        end for
```

```
15        (U, f(U)) ← GROA(U, H[0 … 3n – 1]);
16        if f(U) > f(Yi(t)) then (Xi(t), Yi(t)) ← (Z, U);
17      end for
18      Determine (Xb(t + 1), Yb(t + 1)) by f(Yi(t)) (1 ≤ i ≤ N) in
        P(t + 1) ∪ {(Xb(t), Yb(t))};
19      t ← t + 1;
20    end while
21    return(Yb(t – 1), f(Yb(t – 1))).
```

In HBDE, step 1 is implemented by using the *QuickSort* algorithm in (Cormen et al., 2001). Its time complexity is $O(n\log n)$. The time complexity of both step 2 and step 3–step 6 is $O(N*n)$, since that of GROA is $O(n)$. Because the time complexity of step 8 – step 20 is $O(MaxIt*N*n)$, that of HBDE is $O(n\log n) + O(MaxIt*N*n)$. Since N and $MaxIt$ are linear functions with respect to n, we have $O(MaxIt*N*n) + O(n\log n) = O(n^3)$. HBDE is a stochastic approximation algorithm with polynomial time complexity.

4 Discrete DEs for solving D{0–1}KP

Since the feasible solution is an integer vector in $\{0, 1, 2, 3\}^n$ in the second mathematical model, the standard DE cannot be used directly to solve D{0–1}KP. Therefore, we draw lessons from the idea of HBDE to propose the discrete DE. By using two different encoding conversion functions, which transform the real vector into an integer vector, we propose two discrete DEs, the first discrete differential evolution algorithm (FDDE) and the second discrete differential evolution algorithm (SDDE) for solving D{0–1}KP, separately.

Now, we first introduce the principle of FDDE and its pseudo code description based on the model DE/rand/1/bin.

4.1 FDDE algorithm

Let $X_i = [x_{i0}, x_{i1}, \ldots, x_{i(2n-1)}] \in \mathbf{S}_2$ represent the i^{th} individual of the current population in FDDE, where $S_2 = \prod_{j=1}^{2n}[low, high]$; $low < 0 < high$; both low and $high$ are real numbers; $i = 1, 2, \ldots, N$; N is the population size; n represents the number of item groups of D{0–1}KP.

We note that the two bit binary numbers corresponding to integers 0, 1, 2, 3 are 00, 01, 10 and 11, separately. The '0' and '1' can correspond to the positive and negative. Therefore, we can define the encoding conversion function $W = g_2(V)$ of FDDE as follows:

$$w_j = \begin{cases} 0, & \text{if } v2j < 0 \text{ and } v2j+1 < 0; \\ 1, & \text{if } v2j < 0 \text{ and } v2j+1 \geq 0; \\ 2, & \text{if } v2j < 0 \text{ and } v2j+1 < 0; \\ 3, & \text{if } v2j < 0 \text{ and } v2j+1 \geq 0. \end{cases} \tag{12}$$

Where, $V = [v_0, v_1, \ldots, v_{2n-1}] \in \mathbf{S}_2$ is a $2n$-dimensional real vector and $W = [w_0, w_1, \ldots, w_{n-1}] \in \{0, 1, 2, 3\}^n$ is an n-dimensional integer vector.

Let $Y = [y_0, y_1, \ldots, y_{n-1}] \in \{0, 1, 2, 3\}^n$ represent the integer vector which is obtained by using the encoding conversion function $Y = g_2(X)$, $X \in \mathbf{S}_2$ and X is an individual in FDDE. Then Y is a potential solution to D{0–1}KP corresponding to X. By using NROA (He et al., 2016) to repair and optimise Y, we can make it a feasible solution to D{0–1}KP. $f(Y)$ is calculated as the fitness of X.

For example, suppose the scale of D{0–1}KP instance $\mathbf{I}_2$ is $3n = 15$ and $S_2 = \prod_{j=1}^{10}[-5.0, 5.0]$; $X = [2.13, -0.51, -3.93, -2.77, -3.82, 3.29, 4.12, 1.15, -1.22, -2.35] \in \mathbf{S}_2$ is an individual in FDDE. Then, the feasible solution corresponding to X is $Y = [2, 0, 1, 3, 0] \in \{0, 1, 2, 3\}^5$. The schematic diagram of $Y = g_2(X)$ is shown as Table 2.

For the $i^{th}(i = 1, 2, \ldots, N)$ individual $X_i = [x_0, x_1, \ldots, x_{2n-1}] \in \mathbf{S}_2$ in current population of FDDE, let $Z = [z_0, z_1, \ldots, z_{2n-1}] \in \mathbf{S}_2$ represent a temporary $2n$-dimensional real vector, which is used to derive the offspring individual of X_i in the following operation. The mutation and crossover operations of FDDE are achieved based on formulas (9) and (10) the same as HBDE, so there is no repeat any more. But it should be emphasised that all individual's coding are $2n$-dimensional real vectors in $\mathbf{S}_2$ and the range of index j in formulas (9) and (10) is from 0 to $2n - 1$.

In order to achieve the selection operation in FDDE, first we use the encoding conversion function $U = g_2(Z)$ to obtain an integer vector $U = [u_0, u_1, \ldots, u_{n-1}] \in \{0, 1, 2, 3\}^n$. Because U may not be a feasible solution to D{0–1}KP, we use NROA to repair and optimise U. The objective function value $f(U)$ is considered as the fitness of Z. Then the current individuals X_i or Z are selected based on formula (11).

Table 1 The encoding conversion function $Y = g_1(X)$

X	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{11}
	1.15	–4.73	3.44	–2.32	–0.71	–1.08	2.29	4.11	–3.69	3.15	–4.01
g_1	⇩	⇩	⇩	⇩	⇩	⇩	⇩	⇩	⇩	⇩	⇩
	1	0	1	0	0	0	1	1	0	1	0
Y	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{11}

Table 2 The encoding conversion function $Y = g_2(X)$

X	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
	2.13,	–0.51	–3.93,	–2.77	–3.82,	3.29	4.12,	1.15	–1.22,	–2.35
g_2	⇩		⇩		⇩		⇩		⇩	
	2		0		1		3		0	
Y	y_0		y_1		y_2		y_3		y_4	

By the above exposition, the algorithm principle of FDDE is described as follows:

1 Initialisation: generate the population $\mathbf{P}(0) = \{X_i(0) \in \mathbf{S}_2 | \ 1 \le i \le N)\}$ randomly. Use the encoding conversion function $Y_i(0) = g_2(X_i(0))$ to get the potential solution $Y_i(0) \in \{0, 1, 2, 3\}^n$ corresponding to the individual $X_i(0)$. Repair and optimise $Y_i(0)$ by using NROA. Then based on $f(Y_i(0))$ $(1 \le i \le N)$, determine the current global optimal individual $X_b(0) = [x_{b0}(0), x_{b1}(0), \ldots, x_{b,2n-1}(0)] \in \mathbf{S}_2$ and its corresponding feasible solution $Y_b(0) = [y_{b0}(0), y_{b1}(0), \ldots, y_{b,n-1}(0)] \in \{0, 1, 2, 3\}^n$. Let t be the loop control variable and set $t = 0$.

2 The $(t + 1)^{th}$ iteration evolution in FDDE: for each individual $X_i(t)$ $(1 \le i \le N)$ in population $\mathbf{P}(t)$, firstly we use (9) and (10) to generate a temporary individual $Z \in \mathbf{S}_2$ and get the potential solution $U \in \{0, 1, 2, 3\}^n$ corresponding to Z by using the encoding conversion function $U = g_2(Z)$. Repair and optimise U by using NROA and then calculate the value of $f(U)$. If $f(U) > f(Y_i(t))$, replace $(X_i(t), Y_i(t))$ with (Z, U); otherwise, keep $(X_i(t), Y_i(t))$ unchanged. The new population $\mathbf{P}(t + 1)$ is generated after all the individuals of $\mathbf{P}(t)$ have finished the above operations. In $\mathbf{P}(t + 1) \cup \{(X_b(t), Y_b(t))\}$, determine the global optimal individual $X_b(t + 1) = [x_{b0}(t + 1), x_{b1}(t + 1), \ldots, x_{b,2n-1}(t + 1)] \in \mathbf{S}_2$ and its corresponding feasible solution $Y_b(t + 1) = [y_{b0}(t + 1), y_{b1}(t + 1), \ldots, y_{b,n-1}(t + 1)] \in \{0, 1, 2, 3\}^n$ based on individual fitness. Set $t = t + 1$.

3 Termination determination: if $t \le MaxIt$, go back to (2) to execute the next iteration evolution process; otherwise, output $(Y_b(t - 1), f(Y_b(t - 1)))$ and end the algorithm.

The pseudo-code of the FDDE is described as follows:

Algorithm 2 FDDE

```
Input:  The D{0–1}KP instances, parameters N, MaxIt, low,
        high, F and CR;
Output: Approximate (or optimal) solution Yb(t – 1) and its
        objective function value f(Yb(t – 1)).
H[0 … 3n – 1] ← Sort({pj / wj| pj ∈ P, wj ∈ W, 0 ≤ j ≤
3n – 1});
Generate initial population P(0) = {Xi(0) ∈ S2 | 1 ≤ i ≤ N}
randomly;
    for i = 1 to N do
        Yi(0) ← g2(Xi(0));
        (Yi(0), f(Yi(0))) ← NROA(Yi(0), H[0 … 3n – 1])
    end for
    Determine (Xb(0), Yb(0)) by f(Yi(0)) (1 ≤ i ≤ N) in P(0);
    t ← 0;
    while (t < MaxIt) do
        for i = 1 to N do
            for j = 0 to n – 1 do
                if (r ≤ CR∨j = R(i)) then zj ← yp1,j(t) +
                F(yp2,j(t) – yp3,j(t)) else zj ← yij(t);
                if (zj < low or zj > high) then zj ← rand(0, 1)
                * (high-low) + low;
            end for
            U ← g2(Z);
            (U, f(U)) ← NROA(U, H[0 … 3n – 1]);
            if f(U) > f(Yi(t)) then (Xi(t), Yi(t)) ← (Z, U);
        end for
        Determine (Xb(t + 1), Yb(t + 1)) by f(Yi(t)) (1 ≤ i ≤
        N) in P(t + 1) ∪ {(Xb(t), Yb(t))};
        t ← t + 1;
    end while
    return (Yb(t – 1), f(Yb(t – 1))).
```

Because the time complexity of NROA is $O(n)$, similar to the analysis of HBDE, it is easy to derive the time complexity of FDDE, which is $O(MaxIt*N*n) + O(n\log n) = O(n^3)$. FDDE is also a stochastic approximation algorithm with polynomial time complexity.

4.2 SDDE algorithm

Let $X_i = [x_{i0}, x_{i1}, \ldots, x_{i,n-1}] \in \mathbf{S}_3$ represent the i^{th} $(1 \le i \le N)$ individual in the current population of SDDE, where $S_3 = \prod_{j=1}^{n}[low, high]$, $low < 0 < high$; both *low* and *high* are real numbers; N is the population size; n is the number of item groups in D{0–1}KP.

Inspired by He et al. (2007) and Greenwood (2009), the encoding conversion function $W = g_3(V)$ of SDDE is defined based on dividing interval [*low*, *high*] to four segments, which correspond to 0, 1, 2 and 3, respectively. Its implementation method is shown as follows:

$$w_j = \begin{cases} 0, & \text{if } low \le v_j < left; \\ 1, & \text{if } left \le v_j < 0; \\ 2, & \text{if } 0 \le vj < right; \\ 3, & \text{if } right \le vj \le high. \end{cases} \tag{13}$$

Where, $V = [v_0, v_1, \ldots, v_{n-1}] \in \mathbf{S}_3$ is an n-dimensional real vector; $W = [w_0, w_1, \ldots, w_{n-1}] \in \{0, 1, 2, 3\}^n$ is an n-dimensional integer vector. Both left and right are real numbers and $low < left < 0 < right < high$.

Obviously, [*low*, *high*] = [*low*, *left*) ∪ [*left*, 0) ∪ [0, *right*) ∪ [*right*, *high*]. The intervals [*low*, *left*), [*left*, 0), [0, *right*) and [*right*, *high*] are numbered as 0, 1, 2 and 3 separately. Based on $W = g_3(V)$, the component w_j of vector W is defined as the number of the interval which the component v_j of vector V belongs to. That is, if $v_j \in$ [*low*, *left*), then $w_j = 0$; if $v_j \in [left, 0)$, then $w_j = 1$; if $v_j \in [0, right)$, then $w_j = 2$; if $v_j \in [right, high]$, then $w_j = 3$.

For example, suppose the scale of D{0–1}KP instance I_3 is $3n = 15$. $S_3 = \prod_{j=1}^{5}[-5.0, 5.0]$, $left = -2.5$, $right = 2.5$; $X = [3.14, -1.73, 1.29, 2.71, -3.13] \in \mathbf{S}_3$ is an individual of SDDE, then $[-5.0, 5.0] = [-5.0, -2.5) \cup [-2.5, 0) \cup [0, 2.5) \cup [2.5, 5.0]$ is obtained. Therefore, the potential solution corresponding to X is $Y = [3, 1, 2, 3, 0] \in \{0, 1, 2, 3\}^5$. The schematic diagram of $Y = g^3(X)$ is shown as Table 3.

Table 3 The encoding conversion function $Y = g_3(X)$

X	x_0	x_1	x_2	x_3	x_4
	3.14	–1.73	1.29	2.71	–3.13
g_3	⇓	⇓	⇓	⇓	⇓
	3	1	2	3	0
Y	y_0	y_1	y_2	y_3	y_4

For the i^{th} individual $X_i = [x_0, x_1, \ldots, x_{n-1}] \in \mathbf{S}_3$ of SDDE, let $Z = [z_0, z_1, \ldots, z_{n-1}] \in \mathbf{S}_3$ represent a temporary n-dimensional real vector. The mutation and crossover operations of SDDE based on DE/rand/1/bin mode are achieved by using formulas (9) and (10). It should be noted that the individual encodes involved in the operation are all n-dimensional real vectors in $\mathbf{S}_3$ and the range of index j in formulas (9) and (10) is from 0 to $n - 1$.

In order to achieve the selection operation in SDDE, we first transform the real vector $Z = [z_0, z_1, \ldots, z_{n-1}] \in \mathbf{S}_3$ into an integer vector $U = [u_0, u_1, \ldots, u_{n-1}] \in \{0, 1, 2, 3\}^n$ by using the encoding conversion function $U = g_3(Z)$. Since U may not be a feasible solution to D{0–1}KP, we repair and optimise it by using NROA and calculate its objective function value $f(U)$. Then the current individuals X_i or Z are selected based on formula (11).

The algorithm principle of SDDE is similar to those of HBDE and FDDE. So it is not repeated any more. Then the pseudo-code of SDDE based on the DE/rand/1/bin mode is described as follows:

Algorithm 3 SDDE

Input: The D{0–1}KP instances, parameters N, $MaxIt$, low, $high$, $left$, $right$, F and CR;

Output: Approximate (or optimal)solution $Y_b(t-1)$ and its objective function value $f(Y_b(t-1))$.

H[0 … 3n – 1] ← ***Sort***$(\{p_j / w_j \mid p_j \in P, w_j \in W, 0 \le j \le 3n - 1\})$;
Generate initial population $\mathbf{P}(0) = \{X_i(0) \in \mathbf{S}_3 \mid 1 \le i \le N\}$ randomly;
for $i = 1$ to N do
  $Y_i(0) \leftarrow g_3(X_i(0))$;
  $(Y_i(0), f(Y_i(0))) \leftarrow$ **NROA**$(Y_i(0)$, H[0 … $3n-1$])
end for
Determine $(X_b(0), Y_b(0))$ by $f(Y_i(0))$ $(1 \le i \le N)$ in $\mathbf{P}(0)$; $t \leftarrow 0$;
while $(t \le MaxIt)$ **do**
  for $i = 1$ to N **do**
    for $j = 0$ to $n - 1$ **do**
      if $(r \le CR \vee j = R(i))$ then $z_j \leftarrow y_{p1,j}(t) + F(y_{p2,j}(t) - y_{p3,j}(t))$ **else** $z_j \leftarrow y_{ij}(t)$;
      if ($z_j < low$ or $z_j > high$) **then** $z_j \leftarrow rand(0, 1)*(high-low) + low$;
    end for
    $U \leftarrow g_3(Z)$;
    $(U, f(U)) \leftarrow$ **NROA**$(U$, H[0 … $3n - 1$]);
    if $f(U) > f(Y_i(t))$ **then** $(X_i(t), Y_i(t)) \leftarrow (Z, U)$;
  end for
  Determine $(X_b(t+1), Y_b(t+1))$ by $f(Y_i(t))$ $(1 \le i \le N)$ in $\mathbf{P}(t+1) \cup \{(X_b(t), Y_b(t))\}$;
  $t \leftarrow t + 1$;
end while
return$(Y_b(t-1), f(Y_b(t-1)))$.

Obviously, the time complexity of SDDE is $O(MaxIt*N*n) + O(n\log n) = O(n^3)$. SDDE is also a stochastic approximation algorithm with polynomial time complexity.

5 Computational experiments

The comparison of HBDE, FDDE and SDDE is shown in Table 4. It can be seen that the three algorithms are designed in exactly the same way, which are based on the encoding conversion of real numbers to integer vectors. HBDE converts a real vector to a 0–1 vector and FDDE and SDDE transform a real number vector into an integer vector; HBDE is only applicable to combinatorial optimisation problems with feasible solutions for 0–1 vectors and FDDE and SDDE are suitable for the combinatorial optimisation problem with feasible solutions for integer vectors; for D{0–1}KP problems, HBDE performs much better than FDDE and SDDE; the computing speeds of both FDDE and SDDE are faster than that of HBDE.

In this section, for testing the performances of HBDE, FDDE and SDDE, we use them to solve the four kinds of large scale benchmarked instances (He et al., 2016) of D{0–1}KP and compare the results with those of FirEGA and SecEGA. The microcomputer used is Acer Aspire E1-570G notebook; hardware configuration is Intel(R) Core(TM)i5-3337u CPU-1.8GHz, 4GB DDR3 RAM (3.82GB available); the operating system is Microsoft Windows 8 and programming language is C++ and the compiler environment is Visual C++6.0; Execl 2007 and MATLAB 7.10.0.499 (R2010a) are used to draw the fitting curve of approximate ratio (Du et al., 2012) and the convergence curves of four algorithms, respectively.

Due to the current absence of benchmarks set of D{0–1}KP, in this paper, four types of large scale D{0–1}KP instances proposed in (He et al., 2016) are used to be calculated. They are:

1 Uncorrelated instances of D{0–1}KP named from UDKP1 to UDKP10.

2 Weakly correlated instances of D{0–1}KP named from WDKP 1 to WDKP 10.

3 Strongly correlated instances of D{0–1}KP named from SDKP 1 to SDKP 10.

4 Inverse correlated instances of D{0–1}KP named from IDKP 1 to IDKP 10. Specific examples of various types of data see http://pan.baidu.com/s/1o6MJVEq.

Table 4 A comparison among HBDE, FDDE and SDDE

Comparison objects	*HBDE*	*FDDE*	*SDDE*
Discretisation method	Convert a real vector to a binary vector	Convert a real vector to an integer vector	Convert a real vector to an integer vector
Applicable problems	The combinatorial optimisation problem with feasible solution as binary vector	The combinatorial optimisation problem with feasible solution as integer vector	The combinatorial optimisation problem with feasible solution as integer vector
Running speed	The slowest	Medium	The fastest
Performance of solving D{0-1}KP	The best	Medium	The worst

The population size of HBDE, FDDE and SDDE is $N = 50$ and the iteration number is $MaxIt = 3n$ ($3n$ is the items amount in D{0–1}KP), $low = –5.0$, $high = 5.0$. Furthermore, set $F = 0.2$ and $CR = 0.3$ in HBDE, FDDE and SDDE respectively and $left = –2.5$ and $right = 2.5$ are set in SDDE. The parameter of FirEGA and SecEGA is set the same as He et al. (2016).

For each D{0–1}KP instance, all algorithms are executed independently 100 times. The computing results of each algorithm are given in Tables 5–8, where, columns under the caption the '*opt*' report the optimal value found by the dynamic programming method (referred to as DPDKP); the 'b*est*', 'w*orst*' and 'm*ean*' report the best value, the worst value and the average value found by HBDE, FDDE,SDD and FirEGA among 100 times execution independently; the '*StaDe*' reports the standard deviation of the 100 times execution; the '*Gap* (%)' reports the average gap between the best values (*best*) found by every algorithm and the optimal value (*opt*). This gap is calculated by $Gap(\%) = 100 * (opt – best) / opt$. Because the performance of FirEGA is better than that of SecEGA, the results of SecEGA are shown in Tables 5–8 and Figures 1–3.

The fitting curves of approximation ratio which is defined by *opt/mean* for all algorithms are given in Figures 1–4.

From Table 5, we can see that the *worst* values of HBDE, FDDE and SDDE are all better than the *best* value of FirEGA. Thus, for the UDKP class instances, HBDE, FDDE and SDDE all perform much better than FirEGA and SecEGA.

It can be seen from Table 6 that all the indicators of HBDE are optimal and all its *worst* values are better than the *best* value of FirEGA; the *best*, *mean* and *worst* values of FDDE are better than those of FirEGA, respectively; the Mean and Worst values of SDDE are better than those of FirEGA correspondingly, besides WDKP1. Therefore, for the WDKP class instances, HBDE, FDDE and SDDE all perform much better than FirEGA and SecEGA.

The results in Table 7 show that the *best*, *mean* and *worst* values of both HBDE and SDDE are better than those of FirEGA, respectively. Besides, the three indicators' values of SDKP1, SDKP6 and FDDE are better than those of FirEGA, which indicates that for the SDKP class instances, HBDE, FDDE and SDDE all perform much better than FirEGA and SecEGA.

The results in Table 8 represent that every indicator of HBDE is the optimal, while FDDE, SDDE and FirEGA are similar to each other with respect to each indicator. Thus, for the IDKP class instances, HBDE performs the best among the four algorithms; FDDE, SDDE and FirEGA perform similarly and better than SecEGA.

EA is a kind of stochastic approximation algorithm. It usually uses the off-line performance measure (Kashan et al., 2013) on *mean* as the indicator to compare the convergent performances of different algorithms. In order to compare the convergent performances of HBDE, FDDE, SDDE and FirEGA, we will draw the off-line performance curves on *mean* of the four algorithms.

Figure 1 The fitting curves of *opt/mean* for UDKP instances (see online version for colours)

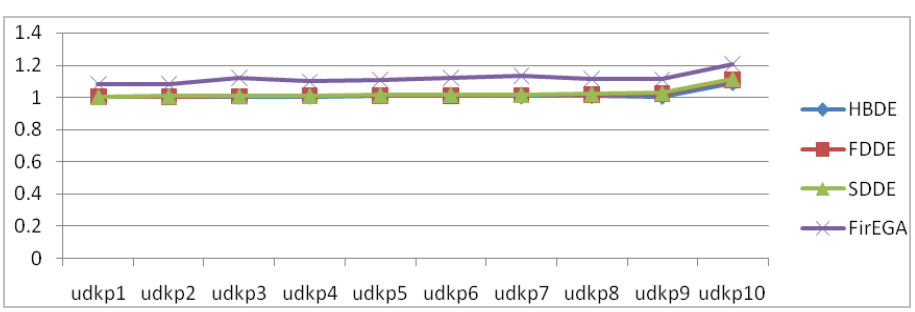


Figure 2 The fitting curves of *opt/mean* for WDKP instances (see online version for colours)

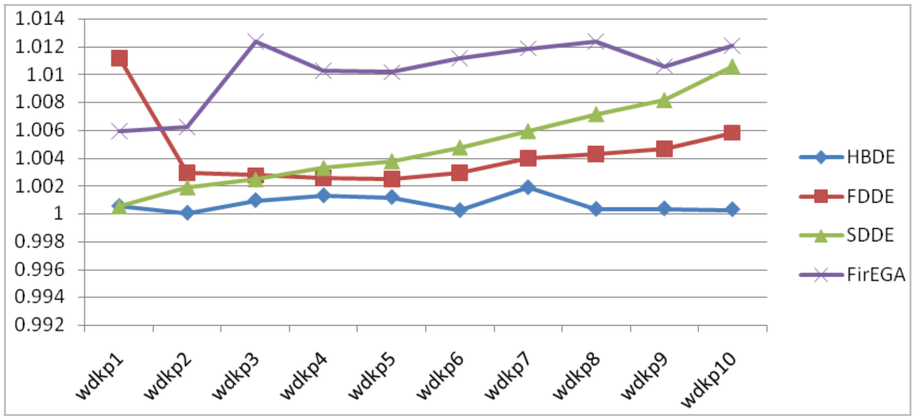


Figure 3 The fitting curves of *opt/mean* for SDKP instances (see online version for colours)

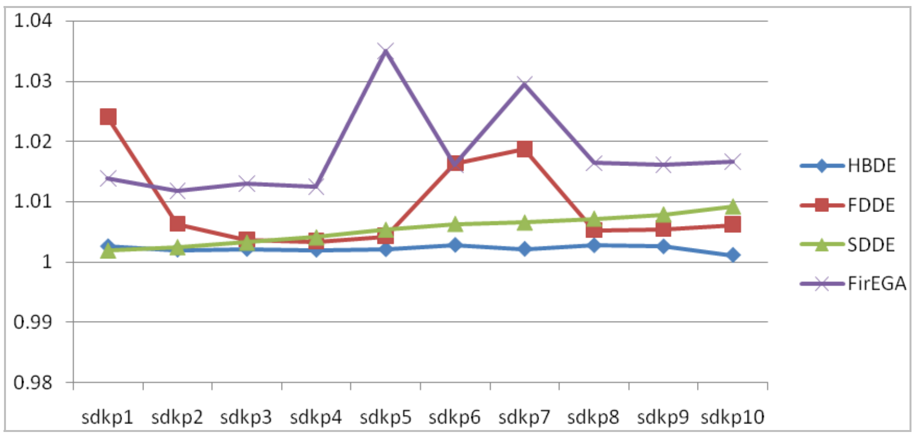


Figure 4 The fitting curves of *opt/mean* for IDKP instances (see online version for colours)

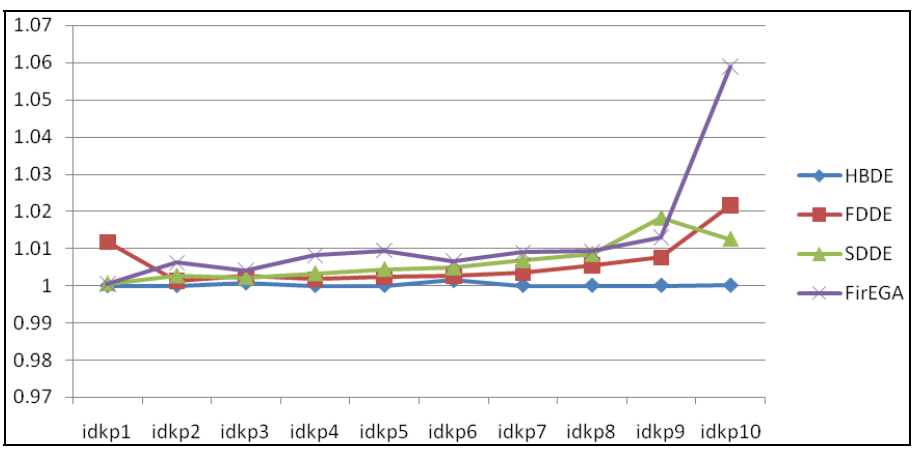


Table 5 Computational results of using the four algorithms to solve UDKP1-UDKP10

Instance	*Opt*	*Algorithm*	*Best*	*Mean*	*Worst*	*StaDe*	*Gap*
UDKP1	85740	HBDE	**85,740**	85,657.9	85,306	85.04	0
		FDDE	85,730	85,521.8	85,211	161.15	0.0117
		SDDE	**85,740**	85,581.6	85,302	129.10	0
		FirEGA	80,650	79,313.1	78,198	731.67	5.9366
UDKP2	163744	HBDE	**163,744**	163,714	163,421	54.99	0
		FDDE	163,554	162,903	161,827	328.29	0.1160
		SDDE	163,519	162,783	161,523	408.44	0.1374
		FirEGA	153,870	151,130	149,649	905.33	6.0301
UDKP3	269393	HBDE	269,125	268,638	267,789	306.07	0.0995
		FDDE	268,780	267,583	266,063	582.68	0.2275
		SDDE	268,679	267,450	265,670	561.96	0.2650
		FirEGA	246,593	240,827	237,780	1,687.92	8.4635
UDKP4	347599	HBDE	347,015	346,381	345,308	344.85	0.1680
		FDDE	346,304	344,909	342,813	693.37	0.3726
		SDDE	345,906	344,418	341,709	832.35	0.4871
		FirEGA	320,572	316,599	313,758	1,446.88	7.7753
UDKP5	442644	HBDE	441,708	440,752	439,752	505.568	0.2115
		FDDE	439,668	437,592	434,087	939.93	0.6723
		SDDE	439,842	436,975	434,136	1,009.29	0.6330
		FirEGA	402,255	398,764	394,716	1,746.24	9.1245
UDKP6	536578	HBDE	535,537	534,315	533,054	539.71	0.1940
		FDDE	532,398	530,101	527,239	1,035.71	0.7790
		SDDE	532,040	528,607	525,306	1,246.84	0.8457
		FirEGA	484,241	478,109	472,852	2,137.30	9.7538
UDKP7	635860	HBDE	634,566	633,229	631,511	627.26	0.2035
		FDDE	631,094	628,040	624,212	1,285.20	0.7495
		SDDE	629,555	626,562	622,660	1,357.25	0.9916
		FirEGA	565,932	560,668	556,327	1,939.30	10.9974
UDKP8	650206	HBDE	648,066	645,904	643,524	804.18	0.3291
		FDDE	642,279	639,885	636,820	1,184.57	1.2192
		SDDE	641,667	638,261	634,451	1,449.75	1.3133
		FirEGA	590,419	584,494	579,453	2,149.00	9.1951
UDKP9	718532	HBDE	717,936	716,798	715,241	488.95	0.0829
		FDDE	707,690	703,814	697,238	1,729.29	1.5089
		SDDE	705,468	700,988	697,097	1,649.67	1.8182
		FirEGA	651,779	646,642	642,409	2,000.70	9.2902
UDKP10	779460	HBDE	717,936	716,798	715,241	488.95	7.8932
		FDDE	707,690	703,814	697,238	1,729.29	9.2077
		SDDE	705,468	700,988	697,097	1,649.67	9.4927
		FirEGA	651,779	646,642	642,409	2,000.70	16.3807

Table 6 Computational results of using the 4 algorithms to solve WDKP1-WDKP10

Instance	*Opt*	*Algorithm*	*Best*	*Mean*	*Worst*	*StaDe*	*Gap*
WDKP1	83,098	HBDE	83,086	83,054.6	82,990	28.94	0.0144
		FDDE	82,801	**82,178.6**	81,428	394.46	0.3574
		SDDE	**83,098**	83,053.1	82,949	38.04	0
		FirEGA	82,750	82,611.1	82,443	97.70	0.4188
WDKP2	138,215	HBDE	**138,215**	138,210	138,155	9.88	0
		FDDE	138,139	137,811	136,403	323.16	0.0550
		SDDE	138,187	137,951	137,640	108.45	0.0203
		FirEGA	137,723	137,360	137,137	114.61	0.3560
WDKP3	256,616	HBDE	256,548	256,373	256,114	89.50	0.0265
		FDDE	256,356	255,916	253,567	416.38	0.1013
		SDDE	256,414	255,979	255,427	194.46	0.0787
		FirEGA	254,240	253,474	253,141	185.76	0.9259
WDKP4	315,657	HBDE	315,469	315,250	314,843	113.76	0.0596
		FDDE	315,248	314,856	314,336	158.81	0.1296
		SDDE	315,073	314,616	314,168	211.75	0.1850
		FirEGA	313,957	312,447	311,577	544.48	0.5386
WDKP5	428,490	HBDE	428,273	427,987	427,522	147.86	0.0506
		FDDE	427,967	427,439	426,877	235.87	0.1221
		SDDE	427,517	426,870	425,603	342.58	0.2271
		FirEGA	425,929	424,176	422,401	907.12	0.5977
WDKP6	466,050	HBDE	466,049	465,947	465,631	90.61	0.0002
		FDDE	465,299	464,681	463,872	290.77	0.1611
		SDDE	464,592	463,833	462,183	424.77	0.3128
		FirEGA	463,586	460,903	456,908	1,794.04	0.5287
WDKP7	547,683	HBDE	547,371	546,656	546,146	219.08	0.0570
		FDDE	546,325	545,514	544,786	348.27	0.2480
		SDDE	545,526	544,438	543,129	494.23	0.3938
		FirEGA	544,371	541,257	536,857	1,695.86	0.6047
WDKP8	576,959	HBDE	576,954	576,776	576,431	108.33	0.0009
		FDDE	575,274	574,493	573,278	399.52	0.2920
		SDDE	574,437	572,847	571,028	612.60	0.4371
		FirEGA	573,448	569,905	560,168	3,128.92	0.6085
WDKP9	650,660	HBDE	650,641	650,431	649,990	131.59	0.0029
		FDDE	648,939	647,640	646,460	473.64	0.2645
		SDDE	646,999	645,383	643,915	642.99	0.5627
		FirEGA	647,419	643,831	627,462	3,090.37	0.4981
WDKP10	678,967	HBDE	678,939	678,770	678,394	107.86	0.0041
		FDDE	676,053	675,050	673,441	498.58	0.4292
		SDDE	673,622	671,844	669,813	801.08	0.7872
		FirEGA	675,558	670,869	648,697	5,542.17	0.5021

Table 7 Computational results obtained by using the 4 algorithms to solve SDKP1-SDKP10

Instance	*Opt*	*Algorithm*	*Best*	*Mean*	*Worst*	*StaDe*	*Gap*
SDKP1	94459	HBDE	94390	94216.8	94022	81.81	0.0730
		FDDE	93072	**92241.5**	91323	433.92	1.4684
		SDDE	94440	94277.6	94048	95.19	0.0201
		FirEGA	93276	93160.8	93024	68.83	1.2524
SDKP2	160805	HBDE	160801	160486	160196	138.90	0.0025
		FDDE	160614	159808	157079	617.85	0.1188
		SDDE	160710	160404	159967	146.60	0.0591
		FirEGA	159156	158927	158724	96.53	1.0255
SDKP3	238248	HBDE	238079	237750	237508	112.33	0.0709
		FDDE	237945	237386	235366	329.60	0.1272
		SDDE	237928	237478	236930	178.74	0.1343
		FirEGA	235432	235185	235003	88.93	1.1820
SDKP4	340027	HBDE	339628	339360	338821	144.29	0.1173
		FDDE	339388	338889	338306	246.73	0.1879
		SDDE	339313	338638	337829	256.66	0.2100
		FirEGA	336440	335826	335497	156.41	1.0549
SDKP5	463033	HBDE	462497	462080	461642	81.30	0.1158
		FDDE	461760	461061	460019	380.29	0.2749
		SDDE	461589	460565	459542	419.67	0.3119
		FirEGA	451969	447361	443852	1966.46	2.3895
SDKP6	466097	HBDE	465827	464803	464085	323.50	0.0579
		FDDE	460414	458636	454923	944.33	1.2193
		SDDE	464038	463172	462054	408.05	0.4418
		FirEGA	459443	458709	458418	187.82	1.4276
SDKP7	620446	HBDE	619706	619133	618629	213.89	0.1193
		FDDE	612681	609081	603513	1506.69	1.2515
		SDDE	617974	616413	615031	505.56	0.3984
		FirEGA	607430	602683	599765	1613.95	2.0978
SDKP8	670697	HBDE	669606	668875	668180	304.19	0.1627
		FDDE	668261	667233	666329	381.79	0.3632
		SDDE	666969	665913	664713	448.87	0.5558
		FirEGA	661344	659864	659182	501.92	1.3945
SDKP9	739121	HBDE	737819	737252	736558	261.60	0.1762
		FDDE	736108	735113	733900	434.44	0.4076
		SDDE	735052	733350	732102	566.31	0.5505
		FirEGA	729075	727378	726746	425.11	1.3592
SDKP10	765317	HBDE	764912	764482	763904	208.81	0.0529
		FDDE	761797	760654	759729	495.40	0.4599
		SDDE	760411	758356	756331	733.93	0.6410
		FirEGA	755309	752782	749396	1398.75	1.3077

Table 8 Computational results obtained by using the 4 algorithms to solve IDKP1-IDKP10

Opt	*Algorithm*	*Best*	*Mean*	*Worst*	*StaDe*	*Gap*
70,106	HBDE	**70,106**	70,105.9	70,101	0.85	0
	FDDE	70,037	69,460.1	68,261	463.60	0.0984
	SDDE	**70,106**	70,065.7	70,001	33.25	0
	FirEGA	**70,106**	70,074.4	70,022	23.41	0
118,268	HBDE	**118,268**	118,264	118,169	13.53	0
	FDDE	118,235	118,119	117,462	125.46	0.0279
	SDDE	118,242	117,952	117,467	191.83	0.0220
	FirEGA	118,040	117,535	117,021	179.35	0.1928
234,804	HBDE	**234,804**	234,629	234,441	75.44	0
	FDDE	234,707	234,215	230,310	742.08	0.0413
	SDDE	234,571	234,281	233,835	142.59	0.0992
	FirEGA	234,607	233,845	233,480	214.82	0.0839
282,591	HBDE	**282,591**	282,575	282,436	31.59	0
	FDDE	282,420	282,102	281,623	175.13	0.0605
	SDDE	282,188	281,665	281,114	211.35	0.1426
	FirEGA	282,269	280,301	278,407	1,050.47	0.1139
335,584	HBDE	**335,584**	335,559	335,195	48.70	0
	FDDE	335,255	334,773	334,051	232.12	0.0980
	SDDE	334,736	334,100	333,386	324.65	0.2527
	FirEGA	334,774	332,425	328,796	1,748.59	0.2414
452,463	HBDE	452,211	451,823	451,349	184.48	0.0557
	FDDE	451,786	451,252	450,373	280.71	0.1496
	SDDE	451,049	450,268	449,018	372.58	0.3125
	FirEGA	451,799	449,511	446,355	1,346.80	0.1468
489,149	HBDE	**489,149**	489,101	488,827	64.24	0
	FDDE	488,190	487,468	485,828	362.06	0.1961
	SDDE	487,349	485,832	484,667	506.27	0.3680
	FirEGA	488,460	484,779	475,214	3,287.98	0.1409
533,841	HBDE	**533,841**	533,789	533,486	66.62	0
	FDDE	532,037	530,944	529,696	503.31	0.3379
	SDDE	530,995	529,336	527,160	634.78	0.5331
	FirEGA	532,091	528,948	513,442	3,495.65	0.3278
528,144	HBDE	**528,144**	528,090	527,776	68.52	0
	FDDE	525,640	524,188	522,564	572.26	0.4741
	SDDE	523,598	518,658	510,621	3,228.41	0.8608
	FirEGA	526,103	521,311	501,038	6,242.29	0.3864
581,244	HBDE	**581,244**	581,174	580,777	87.45	0
	FDDE	574,836	568,976	562,190	3,133.03	1.1025
	SDDE	576,602	574,012	570,323	1,104.25	0.7986
	FirEGA	579,446	548,868	573,401	7,456.98	0.3093

Let $X_b(t)$ be the global optimal individual in the t^{th} iteration when algorithm ***A*** (***A*** is HBDE, FDDE, SDDE or FirEGA) is used to solve instance **I**. $Y_b(t)$ is the solution corresponding to $X_b(t)$, $f_i(Y_b(t))$ is the objective function value of $Y_b(t)$ in the i^{th} ($1 \le i \le 100$) time execution. The off-line performance of algorithm ***A*** for instance **I** is defined by $F(t) = \frac{1}{100}\sum_{i=1}^{30} f_i(Y_b(t)),$ where the values of t are $(k*MaxIt) / 30$, $k = 0, 1, 2, \ldots, 30$. $MaxIt = 3n$ is the iteration number of algorithm ***A***; n is the amount of item groups in instance **I**.

Figure 5 Off-line performance curves of four algorithms for UDKP2 (see online version for colours)

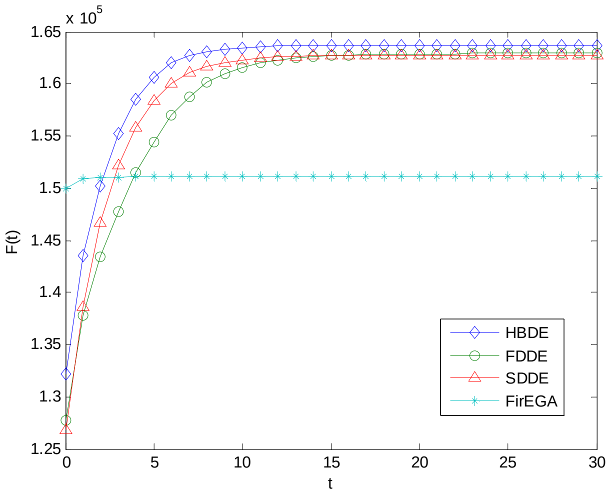


Figure 6 Off-line performance curves of four algorithms forg UDKP5 (see online version for colours)

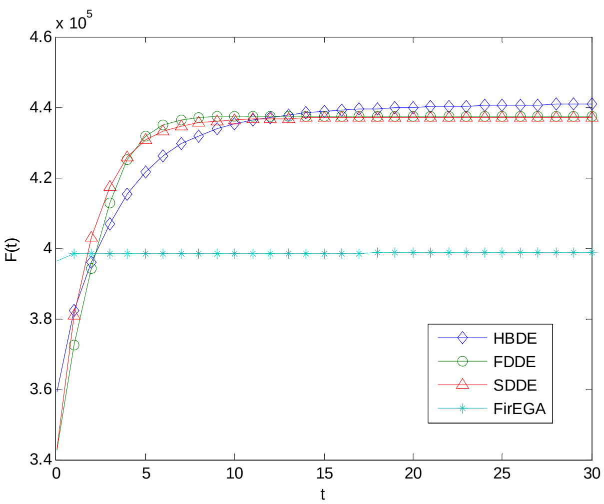


Figure 7 Off-line performance curves of four algorithms for UDKP8 (see online version for colours)

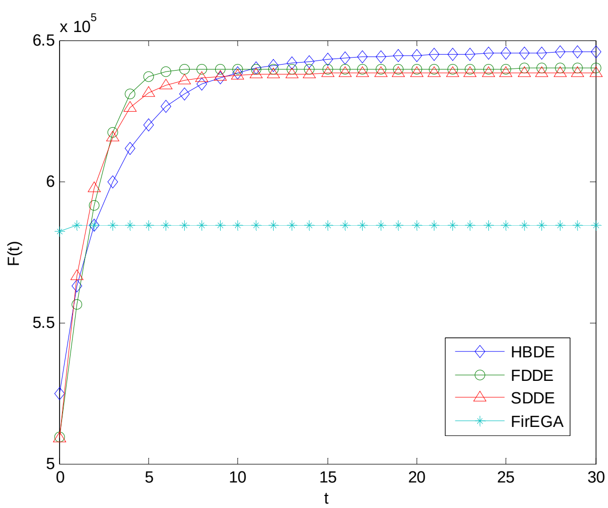


Figure 8 Off-line performance curves of four algorithms for WDKP2 (see online version for colours)

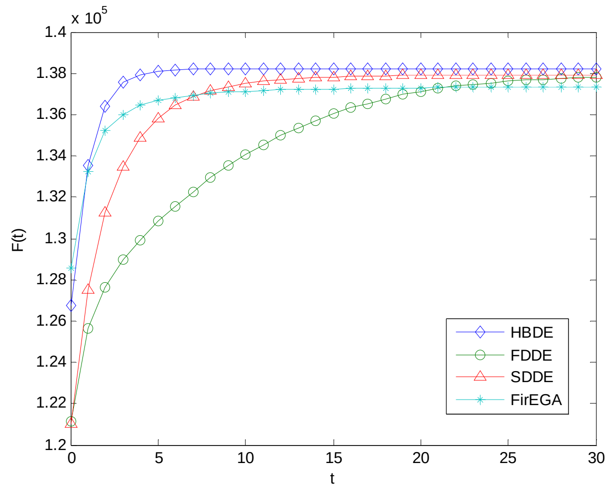


Figure 9 Off-line performance curves of four algorithms for WDKP5 (see online version for colours)

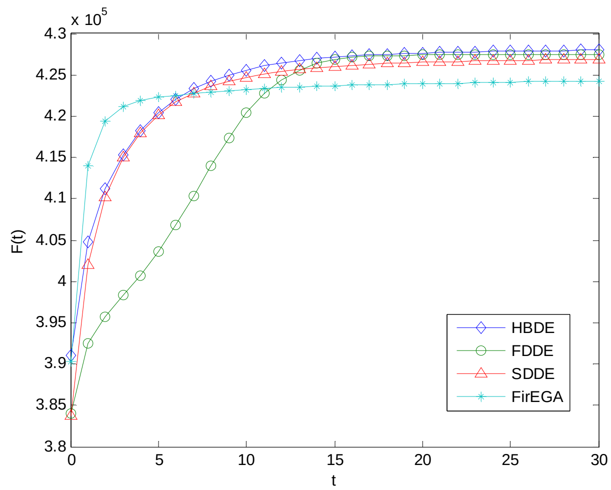


Figure 10 Off-line performance curves of four algorithms for WDKP8 (see online version for colours)

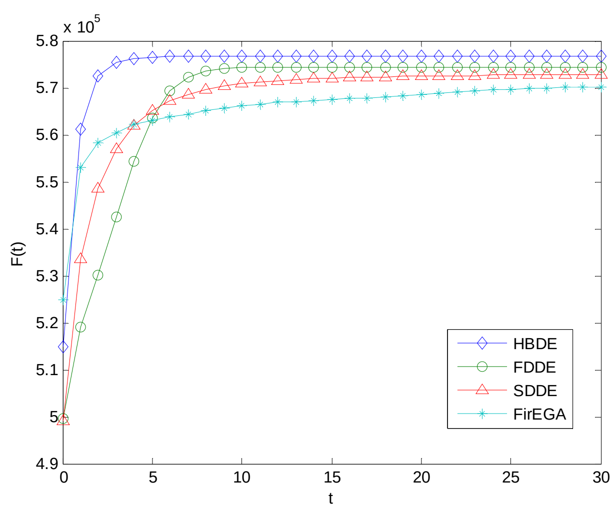


Figure 11 Off-line performance curves of four algorithms for SDKP2 (see online version for colours)

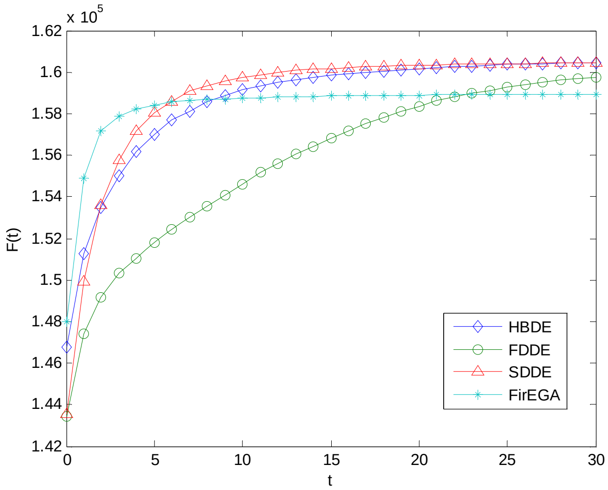


Figure 12 Off-line performance curves of four algorithms for SDKP5 (see online version for colours)

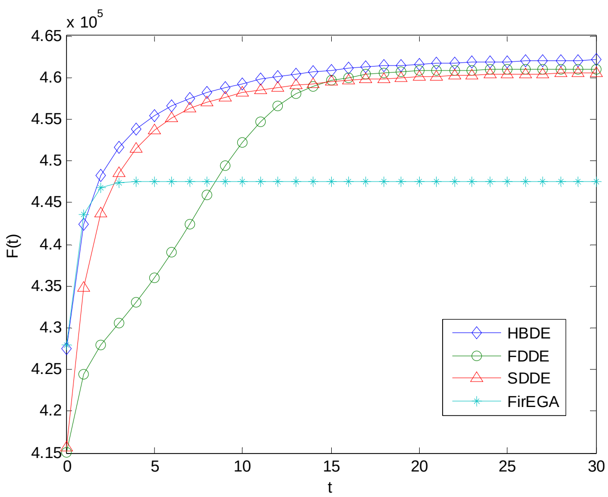


Figure 13 Off-line performance curves of four algorithms for SDKP8 (see online version for colours)

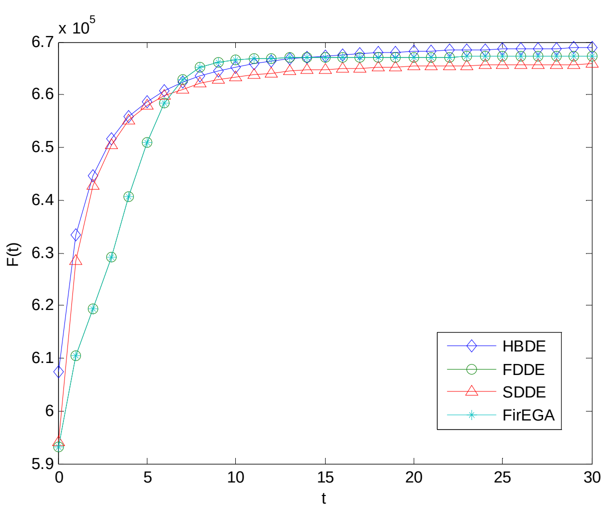


Figure 14 Off-line performance curves of four algorithms for IDKP2 (see online version for colours)

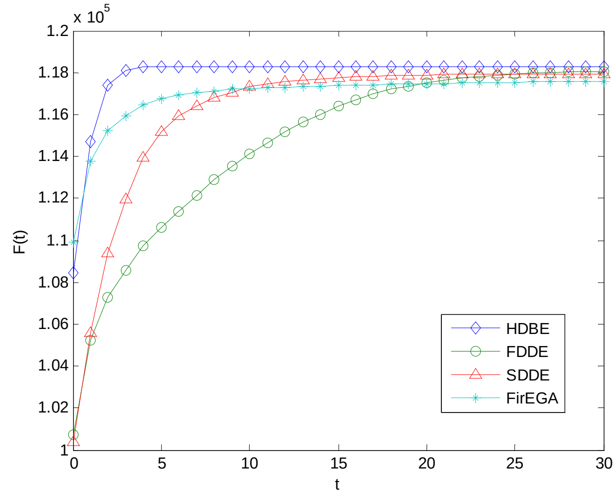


Figure 15 Off-line performance curves of four algorithms for IDKP5 (see online version for colours)

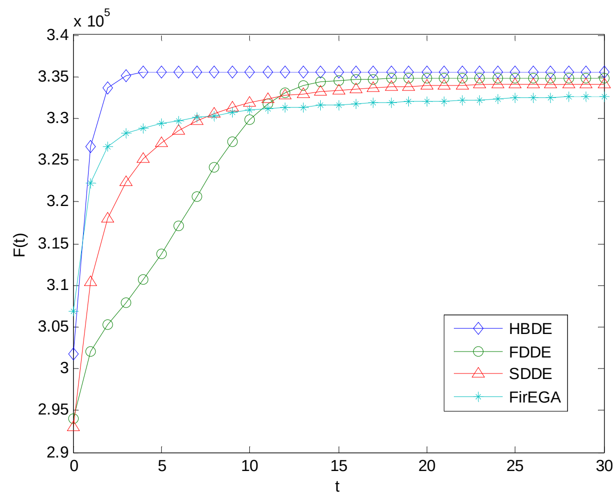
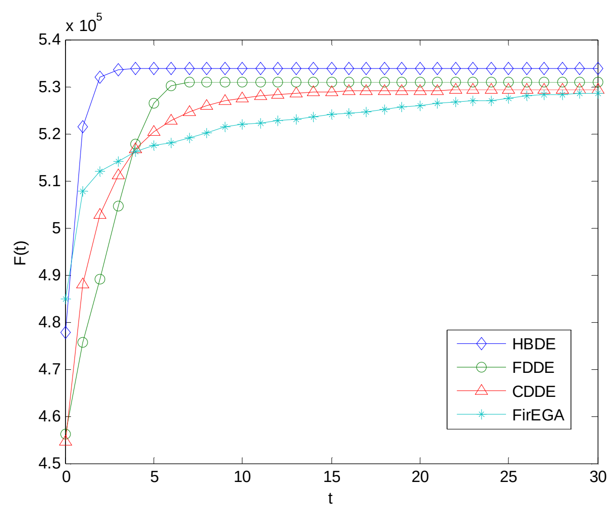


Figure 16 Off-line performance curves of four algorithms for IDKP8 (see online version for colours)



To illustrate succinctly, we only present the off-line performance curves of HBDE, FDDE, SDDE and FirEGA (see from Figure 5–Figure 16) for D{0–1}KP instances with size 3*n* = 600, 1,500 and 2,400.

It can be seen form Figure 5–Figure 16 that the convergence capability of HBDE is the best among those of the four algorithms. For all the D{0–1}KP instances, HBDE can always obtain the best Mean value among the four algorithms after no more than *MaxIt*/2 iterations; FDDE and SDDE can always gain a better *mean* value than FirEGA, although their convergence speeds are not as fast as that of FirEGA at the beginning running of the algorithm; though the convergence speed of FirEGA is very fast at the beginning, it often prematurely gets stuck in local optimums, which makes it cannot perform satisfactorily.

The following conclusions can be drawn based on the above comparison and analysis.

Conclusions: For D{0–1}KP problems, HBDE, FDDE and SDDE perform obviously better than FirEGA and SecEGA, which indicates that compared with the GA, the DE algorithm is more suitable for solving the D {0–1} KP problem. Therefore, it is not only feasible but also very efficient to design a discrete DE algorithm based on the conversion method of converting a real vector into an integer vector.

6 Conclusions

In this paper, DE is used to solve D{0–1}KP. Based on the first and the second mathematical model of D{0–1}KP and the key ideal that makes a real vector transferred to a discrete vector, three discrete DE algorithms, HBDE, FDDE and SDDE, are proposed. By comparing the computational results obtained by using algorithms FirEGA and SecEGA (He et al., 2016) to solve the four kinds benchmark instances of D{0–1}KP, it is illustrated that HBDE, FDDE and SDDE are all suitable for solving D{0–1}KP and HBDE is the best algorithm to solve D{0–1}KP. It is indicated that DE is not only an efficient algorithm for solving D{0–1}KP, but also its discrete methods are highly efficient. In addition, the algorithms proposed in this paper are universal and can be applied to the discretisation of other EAs, such as the fireworks algorithm (FWA) (Tan and Zhu, 2010), fruit fly optimisation (FFO) (Pan, 2012), grey wolf optimiser (GWO) (Mirjalili and Mirjalili, 2014) and artificial algae algorithm (AAA) (Uymaz e tal., 2015).

The history of the D{0–1}KP issue is short and the research achievement is relatively rare. The design of algorithm and construction of benchmarks sets need to be researched further particularly. In addition, for D{0–1}KP instances with profit and weight coefficients distributed in a wide range and with large scale, the exiting exact algorithms are all pseudo-polynomial time. The slow solving speed is an obvious flaw. Therefore, it is necessary to design a fast and efficient algorithm for solving D{0–1}KP. EA will be worth further studying and discussed. More efficient algorithm for D{0–1}KP is needed to be studied further based on other EAs (such as FWA, FFO, GWO and AAA) in the future.

Acknowledgements

The first author and corresponding authors contributed equally the same to this article which was supported by the Macao Science and Technology Development Funds (100/2013/A3 and 081/2015/A3), Basic Research Project of Knowledge Innovation Program in Shenzhen (JCYJ20150324140036825), National Natural Science Foundations of China (61503252 and 71371063), Scientific Research Project Program of Colleges and Universities in Hebei Province (ZD2016005), and Natural Science Foundation of Hebei Province (F2016403055).

References

Abbas, H.A., Sarker, R. and Newton, C. (2001) 'PDE: a pareto-frontier differential evolution approach for multi-objective optimization problems', in *Proceedings of the 2001 Congress on Evolutionary Computation*, IEEE Piscataway, NJ, USA, Vol. 2, pp.971–978, ISBN 0-7803-6657- 3.

Abbass, H. (2002) 'The self-adaptive Pareto differential evolution algorithm', in *Procedings of the IEEE congress on Evolutionary Computation*, IEEE Press, pp.831–836.

Abe, S. (2016) 'Fusing sequential minimal optimization and Newton's method for support vector training', *International Journal of Machine Learning and Cybernetics*, Vol. 7, No. 3, pp.345–364.

Azad, A.K. Rocha, A.M.A.C. and Fernandes, E.M.G.P (2014) 'Asimplified binary artificial fish swarm algorithm for 0–1 quadratic knapsack problems', *Journal of Computer and Applied Mathematics*, Vol. 259, No. 4, pp.897–904.

Azada, A.K., Rochaa, A.M.A.C. and Fernandes, E.M.G.P. (2014) 'A simplified binary artificial fish swarm algorithm for 0-1quadratic knapsack problems', *Journal of Computational and Applied Mathematics*, Vol. 259, No. 4, pp.897–904.

Bäck, T., Fogel, D.B. and Michalewicz, Z. (Eds.) (2007) *Evolutionary Computation2: Advanced Algorithms and Operators*, Institute of Physics, Bristol, UK.

Chih, C-J.M., Chen, M-S. and Ou, T-Y. (2014) 'Particle swarm optimization with time-varying acceleration coefficients for the multidimensional knapsack problem', *Applied Mathematical Modelling*, Vol. 38, No. 4, pp.1338–1350.

Cormen, T.H., Leiserson, C.E., Rivest, R.L. and Stein, C. (2001) *Introduction to Algorithms*, 2nd ed., MIT Press, Cambridge.

Das, S., Abraham, A., Chakraborty, U.K. and Konar, A. (2009) 'Differential evolution using a neighborhood-basedmutation operator', *IEEE Trans. on Evolutionary Computation*, Vol. 13, No.3, pp.526–553.

De, B.P., Kar, R. and Mandal, D. (2015) 'Optimal selection of components value for analog active filter design using simplex particle swarm optimization', *International Journal of Machine Learning and Cybernetics*, Vol. 6, No. 4, pp.621–636.

Dizdar, D., Gershkov, A. and Moldovanu, B. (2011) 'Revenue maximization in the dynamic knapsack problem', *Theoretical Economics*, Vol. 6, No. 2, pp.157–184.

Du, D-Z. and Ko, K-I. (2000) *Theory of Computational Complexity*, Wiley-Interscience, New York.

Du, D-Z., Ko, K-I. and Hu, X (2012) *Design and Analysis of Approximation Algorithms*, Springer Science Business Media LLC, Berlin.

Ekbal, A. and Saha, S. (2016) 'Simultaneous feature and parameter selection using multiobjective optimization: application to named entity recognition', *International Journal of Machine Learning and Cybernetics*, Vol. 7, No. 4, pp.597–611.

Fan, H.Y. and Lampinen, J. (2003) 'A trigonometric mutationoperation to differential evolution', *J. of Global Optimization*, Vol. 27, No. 1, pp.105–129.

Goldberg, D.E. and Smith, R.E. (1987) 'Nonstationary function optimization using genetic algorithms with dominance and diploidy', *International Conference on Genetic Algorithms*, L. Erlbaum Associates Inc., Hillsdale, pp.59–68.

Greenwood, G.W. (2009) 'Using differential evolution for a subclass of graph theory problems', *IEEE Trans. on Evolutionary Computation*, Vol. 13, No. 2, pp.1190–1192.

Guldan, B. (2007) *Heuristic and Exact Algorithms for Discounted Knapsack Problems*, Master thesis, University of Erlangen-Nürnberg, Germany.

Guoming, L., Yuan, D. and Yang, S. (2014) 'A new hybrid combinatorial genetic algorithm for multidimensional knapsack problems', *J. Supercomput.*, Vol. 70, No. 2, pp.930–945.

Hadad, B.S. and Eick, C.F. (1997) 'Supporting polyploidy in genetic algorithms using dominance vectors', in *Proceeding of the Sixth International Conference on Evolutionary Computation*, pp.223–234.

Haddar, B., Khemakhem, M., Hanafi, S. and Wilbaut, C. (2016) 'A hybrid quantum particle swarm optimization for the multidimensional knapsack problem', *Engineering Applications of Artificial Intelligence*, Vol. 55,, No. C, pp.1–13.

He, X. and Han, L. (2007) 'A novel binary differential evolution algorithm based on artificial immune system', in *IEEE Congress on Evolutionary Computation (CEC2007)*, Vol. 2, pp.67–272.

He, Y., Zhang, X., Li, W. et al. (2016) 'Algorithms for randomized time-varying knapsack problems', *Journal of Combinatorial Optimization*, Vol. 31, No. 1, pp.95–117.

He, Y., Wang, X., Kou, Y. (2007) 'A binary differential evolution algorithm with hybrid encoding', *Journal of Computer Research and Development*, Vol. 44, No. 9, pp.1476–1484, in Chinese.

He, Y.C., Wang, X.Z., Li, W.B. and Zhao, S.L. (2017) 'Exact algorithms and evolutionary algorithms for randomized time-varying knapsack problem', *Journal of Software*, Vol. 28, No. 2, pp.185–202, in Chinese [online] http://www.jos.org.cn/1000-9825/4937.htm (accessed 31 July 2017).

He, Y.C., Wang, X.Z., Liu, K.Q. and Wang, Y.Q. (2010) 'Convergent analysis and algorithmic improvement of differentialevolution', *Journal of Software*, Vol. 21, No. 5, pp.875–885, in Chinese.

He, Y-C., Wang, X.-Z., Li, W-B., et al. (2016) 'Research on genetic algorithms for the discounted {0-1} knapsack problem', *Chinese Journal of Computers*, Vol. 39, No. 12, pp.2614–2630.

Kaelo, P. and Ali, M.M. (2006) 'A numerical study of some modifieddifferential evolution algorithms', *European J. of Operational Research*, Vol. 169, No. 3, pp.1176–1184.

Kashan, A.H., Kashan, M.H. and Karimiyan, S. (2013) 'A particle swarm optimizer for grouping problems', *Information Sciences*, Vol. 252, No. 17, pp.81–95.

Kellerer, H., Pferschy, U. and Pisinger, D. (2004) *Knapsack Problems*, Springer, Berlin.

Kong, M., Tian, P. and Kao, Y. (2008) 'A new ant colony optimization algorithm for the multidimensional knapsack problem', *Computer & Operations Research*, Vol. 35, No. 8, pp.2672–2683.

Kulkarni, A.J. (2016) 'Solving 0–1 knapsack problem using cohort intelligence algorithm', *International Journal of Machine Learning and Cybernetics*, Vol. 7, No. 3, pp.427–441.

Lin, G., Zhu, W. and Ali, M.M. (2011) 'An exact algorithm for the 0–1 knapsack problem with a single continuous variable', *J. Glob. Optim.*, Vol. 50, No. 4, pp.657–673.

Lin, G.Y., Lu, Y. and Yao, D.D. (2008) 'The stochastic knapsack revisited: switch-over policies and dynamic pricing', *Operations Research*, Vol. 56, No. 4, pp.945–957.

Marchand, H. and Wolsey, L.A. (1999) 'The 0–1 knapsack problem with a single continuous variable', *Math Program*, Vol. 85, No. 1, pp.15–33.

Martínez-Soto, R. and Castillo, O. (2015) 'A hybrid optimization method with PSO and GA to automatically design Type-1 and Type-2 fuzzy logic controllers', *International Journal of Machine Learning and Cybernetics*, Vol. 6, No. 2, pp.175–196.

Michael, T. (2002) *Heath. Scientific Computing: A Introductory Survey*, McGraw-Hill Companies, Inc., New York.

Mirjalili, S. and Mirjalili, S.M. (2014) 'Andrew Lewis. Grey wolf optimizer', *Advances in Engineering Software*, Vol. 69, pp.46–61.

Nearchou, A.C. and Omirou, S.L. (2006) 'Differential evolution for sequencing and scheduling optimization', *Journal of Heuristics*, Vol. 12, No. 6, pp.395–411.

Noman, N. and Iba, H. (2008) 'Accelerating differential evolution using an adaptive local search', *IEEE Trans. on Evolutionary Computation*, Vol. 12, No. 1, pp.107–125.

Pan, W-T. (2012) 'A new fruit fly optimization algorithm: taking the financial distress model as an example', *Knowledge-Based Systems*, Vol. 26, No. 2, pp.69–74.

Qin, A.K., Huang, V.L. and Suganthan, P.N. (2009) 'Differential evolution algorithm with strategy adaptation for global numerical optimization', *IEEE Trans. on Evolutionary Computation*, Vol. 13, No. 2, pp.398–417.

Rong, A., Figueira, J.R. and Klamroth, K. (2012) 'Dynamic programming based algorithms for the discounted {0–1} knapsack problem', *Applied Mathematics and Computation*, Vol. 218, No. 12, pp.6921–6933.

Sarkar, B., Sarkar, S. and Yu, W.Y. (2016) 'Retailer's optimal strategy for fixed lifetime products', *International Journal of Machine Learning and Cybernetics*, Vol. 7, No. 1, pp.121–133.

Souravlias, D. and Parsopoulos, K.E. (2016) 'Particle swarm optimization with neighborhood-based budget allocation', *International Journal of Machine Learning and Cybernetics*, Vol. 7, No. 3, pp.451–477.

Storn, R. and Price, K.V. (1997q) 'Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces', *J. Global Optimization*, Vol. 11, No. 4, pp.341–359.

Storn, R. and Price, K.V. (1997b) *Differential Evolution – A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces*', Tech. Report TR-95-012, 1995, Institute of Company Secretaries of India, Chennai, Tamil Nadu.

Storn, R., Price, K.V. and Lampinen, J. (2005) *Differential Evolution – A Practical Approach to Global Optimization*, Springer-Verlag, Berlin, Germany.

Sun, G. and Shen, J. (2016) 'Towards organizing smart collaboration and enhancing teamwork performance: a GA-supported system oriented to mobile learning through cloud-based online course', *International Journal of Machine Learning and Cybernetics*, Vol. 7, No. 3, pp.391–409.

Tan, Y. and Zhu, Y. (2010) *Fireworks Algorithm for Optimization*, in Tan, Y., Shi, Y. and Tan, K.C. (Eds), ICSI, LNCS, Springer, Heidelberg, Vol. 6145, p. 355–64.

Tian, N., Ji, Z. and Lai, C-H. (2015) 'Simultaneous estimation of nonlinear parameters in parabolic partial differential equation using quantum-behaved particle swarm optimization with Gaussian mutation', *International Journal of Machine Learning and Cybernetics*, Vol. 6, No. 2, pp.307–318.

Uymaz, S.A., Tezel, G. and Yel, E. (2015) 'Artificial algae algorithm (AAA) for nonlinear global optimization', *Applied Soft Computing*, Vol. 31, No. C, pp.153–171.

Wang, Y., Cai, Z. and Zhang, Q. (2011) 'Differential evolution with composite trial vector generation strategies and control parameters', *IEEE Trans. on Evolutionary Computation*, Vol. 15, No. 1, pp.55–66.

Yao, X., Liu, Y. and Liu, G. (1999) 'Evolutionary programming made faster', *IEEE Trans. Evol. Comput.*, Vol. 3, No. 2, pp.82–102.

Zhang, G.J., He, Y.J., Guo, H.F., Feng, Y.J., Xu, J.M. (2013) 'Differential evolution algorithm for multimodal optimization basedon abstract convex underestimation', *Journal of Software*, Vol. 24, No. 6, pp.1177–1195, in Chinese.

Zhang, H., Song, S. and Zhou, A. (2015) 'A multiobjective cellular genetic algorithm based on 3D structure and cosine crowding measurement', *International Journal of Machine Learning and Cybernetics*, Vol. 6, No. 3, pp.487–500.

Zhao, C. and Li, X. (2014) 'Approximation algorithms on 0-1 linear knapsack problem with a single continuous variable', *J. Comb. Optim.*, Vol. 28, No. 4, pp.910–916.

Appendix 1

GROA algorithm

Let $Flag[0 \ldots n-1]$ be a Boolean array used to note whether there is an item of the item group i that has been put into the knapsack. When $Flag[i] = 1$, there is exactly one item in the knapsack; when $Flag[i] = 0$, there is no item of item group j put into the knapsack. Suppose both $X = [x_0, x_1, \ldots, x_{3n-1}]$ and $Y = [y_0, y_1, \ldots, y_{3n-1}]$ are binary vectors in $[0, 1]^{3n}$. The pseudo-code of algorithm GROA is described as follows:

Algorithm 4 GROA

Input: $X = [x_0, x_1, \ldots, x_{3n-1}] \in [0, 1]^{3n}$ and array H[0 … $3n-1$];

Output: Binary vector $Y = [y_0, y_1, \ldots, y_{3n-1}]$ and its objective function value $f(Y)$.

for $i \leftarrow 0$ to $3n-1$ **do** $y_i \leftarrow 0$;
for $i \leftarrow 0$ to $n-1$ do $Flag[i] \leftarrow 0$;
$fweight \leftarrow 0$; $fvalue \leftarrow 0$; $i \leftarrow 0$;
while ($fweight < C \wedge i \leq 3n-1$) **do**
  if ($x_{H[i]} = 1$) $\wedge$ ($fweight + w_{H[i]} \leq C$) $\wedge$ ($Flag[\lfloor H[i] / 3\rfloor] = 0$) **then**
    $fweight \leftarrow fweight + w_{H[i]}$;
    $y_{H[i]} \leftarrow 1$; $Flag[\lfloor H[i] \rightarrow 1$;
  end if
  $i \leftarrow i + 1$;
end while
for $i \leftarrow 0$ to $3n-1$ **do**
if ($fweight + w_{H[i]} \leq C$) $\wedge$ ($Flag[\lfloor H[i] / 3\rfloor] = 0$) **then**
    $fweight \leftarrow fweight + w_{H[i]}$;
    $y_{H[i]} \leftarrow 1$; $Flag[\lfloor H[i] \leftarrow 1$;
  end if
end for
for $i \leftarrow 0$ to $3n-1$ **do** $fvalue \leftarrow fvalue + y_k * p_k$;
return (Y, $fvalue$).

Source: He et al. (2016)

The output *fvalue* of algorithm GROA is the value of $f(Y)$. That is the feasible solution Y corresponds to the profit sum of the items that have been put into the knapsack. Obviously, the time complexity of algorithm GROA is $O(n)$.

Appendix 2

NROA (He et al., 2016)

The pseudo-code of NROA algorithm is described as follows:

Algorithm 5 NROA

Input: Individual $X = [x_0, x_1, \ldots, x_{n-1}] \in \{0, 1, 2, 3\}^n$ and array H[0 … $3n-1$];

Output: $X = [x_0, x_1, \ldots, x_{n-1}]$ after repaired and optimised and its objective function value $f(X)$.

$fweight \leftarrow 0$; $fvalue \leftarrow 0$; $k \leftarrow 0$;
$temp \leftarrow \sum_{i=0}^{n-1} \lceil x_i / 3 \rceil w_{|3i+(x_i-1)|}$
if $temp > C$ **then**
  while ($fweight < C$) $\wedge$ ($k \leq 3n-1$) **do**
    if ($x_{\lceil H[k]/3 \rceil} = H[k](\text{mod}3) + 1$)) $\wedge$ ($fweight + w_{H[k]} \leq C$) **then** $fweight \leftarrow fweight + w_{H[k]}$;
    else if $x_{\lceil H[k]/3 \rceil} = H[k](\text{mod}3) + 1$) **then** $x_{\lceil H[k]/3 \rceil} \leftarrow 0$;

```
            k ← k + 1;
        end while
    else fweight ← temp;
for i ← 0 to 3n – 1 do
if (x⌈H[k]/3⌉ = 0) ∧ (fweight + wH[k] ≤ C) then
            x⌈H[k]/3⌉ ← H[k](mod3) + 1; fweight ← fweight + wH[k];
    end if
end for
fvalue ← ∑_{i=0}^{n−1} ⌈x_i / 3⌉ p_{|3i+(x_i −1)|};
return (X, fvalue).
```

The output *fvalue* of NROA algorithm is the value of $f(X)$. That is the feasible solution X corresponds to the profit sum of the items that have been put into the knapsack. Obviously, the time complexity of algorithm NROA is $O(n)$.