Noname manuscript No.
(will be inserted by the editor)

Solving Large 0–1 Multidimensional Knapsack Problems by a New Simplified Binary Artificial Fish Swarm Algorithm

Md. Abul Kalam Azad · Ana Maria A.C. Rocha · Edite M.G.P. Fernandes

Received: date / Accepted: date

Abstract The artificial fish swarm algorithm has recently been emerged in continuous global optimization. It uses points of a population in space to identify the position of fish in the school. Many real-world optimization problems are described by 0–1 multidimensional knapsack problems that are NP-hard. In the last decades several exact as well as heuristic methods have been proposed for solving these problems. In this paper, a new simplified binary version of the artificial fish swarm algorithm is presented, where a point/fish is represented by a binary string of 0/1 bits. Trial points are created by using crossover and mutation in the different fish behavior that are randomly selected by using two user defined probability values. In order to make the points feasible the presented algorithm uses a random heuristic drop_item procedure followed by an add_item procedure aiming to increase the profit throughout the adding of more items in the knapsack. A cyclic reinitialization of 50% of the population, and a simple local search that allows the progress of a small percentage of points towards optimality and after that refines the best point in the population greatly improve the quality of the solutions. The presented method is tested on a set of benchmark instances and a comparison with other methods available in literature is shown. The comparison shows that the proposed method can be an alternative method for solving these problems.

Keywords artificial fish swarm · heuristic search · 0–1 knapsack problem · multidimensional knapsack

M.A.K. Azad
Algoritmi Research Centre
E-mail: akazad@outlook.com

A.M.A.C. Rocha
Department of Production and Systems,
Algoritmi Research Centre,
University of Minho, Campus de Gualtar, 4710-057 Braga, Portugal
Tel.: +351 253604740; fax: +351 253604741.
E-mail: arocha@dps.uminho.pt

E.M.G.P. Fernandes
Algoritmi Research Centre
E-mail: emgpf@dps.uminho.pt

Noname manuscript No.
(will be inserted by the editor)

Solving Large 0–1 Multidimensional Knapsack Problems by a New Simplified Binary Artificial Fish Swarm Algorithm

Received: date / Accepted: date

Abstract The artificial fish swarm algorithm has recently been emerged in continuous global optimization. It uses points of a population in space to identify the position of fish in the school. Many real-world optimization problems are described by 0–1 multidimensional knapsack problems that are NP-hard. In the last decades several exact as well as heuristic methods have been proposed for solving these problems. In this paper, a new simplified binary version of the artificial fish swarm algorithm is presented, where a point/fish is represented by a binary string of 0/1 bits. Trial points are created by using crossover and mutation in the different fish behavior that are randomly selected by using two user defined probability values. In order to make the points feasible the presented algorithm uses a random heuristic drop_item procedure followed by an add_item procedure aiming to increase the profit throughout the adding of more items in the knapsack. A cyclic reinitialization of 50% of the population, and a simple local search that allows the progress of a small percentage of points towards optimality and after that refines the best point in the population greatly improve the quality of the solutions. The presented method is tested on a set of benchmark instances and a comparison with other methods available in literature is shown. The comparison shows that the proposed method can be an alternative method for solving these problems.

Keywords artificial fish swarm · heuristic search · 0–1 knapsack problem · multidimensional knapsack

1 Introduction

The artificial fish swarm algorithm (AFSA) that simulates the behavior of a fish school inside water was recently designed and applied in an engineering context [20,21,38,39]. A state-of-the-art regarding hybridizations and applications of AFSA appears in [26]. Fishes desire to stay close to the school to protect themselves from predators and to look for food, and to avoid collisions within the group. The artificial fish is a fictitious entity of a true fish. When applied to an optimization problem, a 'fish' within the school is represented by a point, also known as a candidate solution, and the school is the so-called population, or set of points or solutions. Inspired by fish school behavior, researchers have developed numerical algorithms aiming to converge to a global optimal solution of the optimization problem, in an efficient manner. The environment in which the artificial fish moves, searching for the optimum, is the feasible search space of the problem. The behaviors that a fish swarm exhibits when looking for food inside water are the following: random behavior, chasing behavior, swarming behavior, searching behavior and leaping behavior. A new fish swarm heuristic which gives priority to the chasing behavior in detriment of the swarming one was proposed in a continuous box constrained global optimization context [31]. In the sequence of this work, Rocha

et al. [32] developed an augmented Lagrangian fish swarm based method for globally solving a nonlinear general constrained problem. Improved results were obtained with a novel hyperbolic augmented Lagrangian paradigm [11].

The 0–1 multidimensional knapsack problem (MKP) is a NP-hard combinatorial optimization problem that arises in many practical problems, such as capital budgeting and project selection problem [27,40], allocating processors and databases in a distributed computer system [17], project selection, cargo loading and so on [35]. The 0–1 MKP is formulated as follows:

$$\begin{array}{rl} \text{maximize} & z(\mathbf{x}) \equiv \mathbf{cx} \\ \text{subject to} & \mathbf{Ax} \leq \mathbf{b} \\ & x_j \in \{0,1\}, \quad j = 1,2,\ldots,n, \end{array} \tag{1}$$

where $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ is an n-dimensional row vector of profits, $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ is an n-dimensional column vector of 0–1 decision variables, $\mathbf{A} = [a_{k,j}]$, $k = 1,2,\ldots,m$, $j = 1,2,\ldots,n$ is an $m \times n$ coefficient matrix of resources and $\mathbf{b} = (b_1, b_2, \ldots, b_m)^T$ is an m-dimensional column vector of resource capacities. It should be noted here that, in a 0–1 multidimensional knapsack problem, each element of $\mathbf{c}$, $\mathbf{A}$ and $\mathbf{b}$ is assumed to be nonnegative. The goal of the 0–1 MKP is to find a subset of n items that yields maximum profit z without exceeding resource capacities $\mathbf{b}$.

Several exact as well as heuristic methods have been developed for solving the 0–1 MKP. Exact methods include dynamic programming methods [5,40], branch-and-bound algorithms [16, 17,35], the Fourier-Motzkin elimination based enumeration algorithms [9], asymptotic analysis method [34], statistical analysis method [14], linked LP-relaxations, disjunctive cuts and implicit enumeration [36], core concept based on LP-relaxation [30] and so on. Pisinger [29] has proposed several exact algorithms for solving knapsack problems in his doctoral thesis.

The exact methods are suitable for small dimensional problems. However, when the dimension increases, they cannot solve the problems within a reasonable time period. This is the main motivation to develop stochastic methods and heuristics for solving the MKP. In the context of constrained problems, the penalty functions are used where a penalty term is added to the objective function aiming to penalize constraint violation. But the performance of penalty-type method is not always satisfactory due to the choice of appropriate penalty parameter values. Hence, other alternative constraint handling techniques have emerged in the last decades.

Several heuristic solution methods for solving the 0–1 MKP have appeared in the literature. Hanafi and Fréville [18] proposed a tabu search approach for the 0–1 MKP using the surrogate constraints information. Chu and Beasley [10] proposed the most successful genetic algorithm (GA) for solving the 0–1 MKP. The authors present the drop-add repair operator based on the pseudo-utility ratios in order to make the solutions feasible. Vasquez and Vimont in [37] presented a hybrid method that combines the linear programming with an efficient tabu search. A binary ant colony optimization, called binary ant system (BAS) algorithm based on the drop-add repair operator [10] is provided in [23]. Bonyadi and Li [7] presented a discrete electromagnetism-like mechanism (DEM) by using the drop-add repair operator for solving the 0–1 MKP. Recently a set-based particle swarm optimization (SBPSO) algorithm has been appeared in [24]. The authors used the penalty function method for handling the constraints. Drexl [13] proposed a simulated annealing based on the add-interchange-drop technique for handling the constraints. Sakawa and Kato [33] introduced a genetic algorithm with double strings based on a decoding algorithm. Some other heuristics are available in the literature [1,6,8,12,19,28]. An interesting review of different solution methods for solving the 0–1 MKP is found in [15]. The focus of the paper is on the theoretical properties and contains an overview of approximate and exact solution methods.

Based on AFSA for continuous global optimization, a preliminary binary version of the artificial fish swarm algorithm (bAFSA) was proposed in [3]. The algorithm was tested on a small set of 0–1 MKP problems. However, bAFSA could not solve large dimensional problems satisfactorily in terms of computational effort. In order to create the trial points from the current ones in a population, bAFSA chooses each point/fish behavior according to the number of points inside its 'visual scope', i.e., inside a closed neighborhood centered at the point. To identify those points, the Hamming distance between pairs of points is used. Uniform crossover and mutation are used

in order to create trial points. In order to handle the knapsack constraints, the decoding algorithm proposed in [33] is used to make the points feasible.

To overcome the drawbacks of bAFSA, a simplified binary version of the artificial fish swarm algorithm for solving the 0–1 quadratic knapsack problems is proposed in [4]. The algorithm, therein denoted by S-bAFSA, simplifies the procedures that are used to select the behavior that should be performed to each current point in order to create the corresponding trial point. The selection of each fish/point behavior depends solely on two target probability values. Thus, in S-bAFSA, all the Hamming distances between pairs of points, as well as the comparison between the Hamming distance and the radius of the 'visual scope', are not required. The main goal is to reduce the computational requirements to reach the optimal solution, mainly in terms of execution time.

The purpose of this paper is threefold:

- to preserve the previously tested simplified procedures that aim to select the behaviors that are performed on points of the current population, so that computational burden is kept low;
- to develop a new simplified binary version of AFSA, henceforth denoted by newS-bAFSA, by incorporating new strategies into the fish behaviors and designing a local search aiming to enhance the quality of the solutions;
- to carry out a computational study by extending its use to solve large 0–1 MKP (1) and to compare with other heuristic methods.

In newS-bAFSA, for all points of the population, except the best, the random, chasing and searching behaviors are randomly chosen using two user defined target probability values $0 \leq \tau_1 \leq \tau_2 \leq 1$. Thereafter, an effect-based crossover is used to create the trial points. A simple 4 flip-bit mutation is operated on the best point of the population to create the corresponding trial point. In order to make the points feasible the newS-bAFSA uses a random heuristic drop_item procedure followed by an add_item procedure aiming to increase the profit throughout the adding of more items in the knapsack. Furthermore, to improve the quality of the solutions obtained by the algorithm, a local search based on a flip-bit mutation operated on a pre-defined number of points, with a pre-specified probability, followed by a refinement procedure carried out on the best point alone, is implemented. We also adopt a cyclic reinitialization of 50% of the population to enlarge the exploration properties of the algorithm.

A benchmark set of large 0–1 multidimensional knapsack problems is used to test the performance of the newS-bAFSA. Although the proposal is very simple and easy-to-implement, the comparisons carried out show that the algorithm is a competitive alternative to other heuristic methods from the literature.

The organization of this paper is as follows. We briefly describe the artificial fish swarm algorithm in Section 2. In Section 3 the proposed new simplified binary artificial fish swarm algorithm is outlined. Section 4 describes the experimental results and finally we draw the conclusions of this study in Section 5.

2 Artificial Fish Swarm Algorithm

In this section, we give a brief description of AFSA proposed in [31] for box constrained global optimization problems of type $\text{minimize}_{\mathbf{x}\in\Omega}\ f(\mathbf{x})$. Here $f : \mathbb{R}^n \to \mathbb{R}$ is a nonlinear function that is to be minimized and $\Omega = \{\mathbf{x} \in \mathbb{R}^n : l_j \leq x_j \leq u_j, j = 1, 2, \ldots, n\}$ is the search space. l_j and u_j are the lower and upper bounds of x_j, respectively, and n is the number of variables of the optimization problem.

AFSA works with a population of N points $\mathbf{x}^i, i = 1, 2, \ldots, N$ to identify promising regions looking for a global solution [38]. $\mathbf{x}^i$ is a floating-point encoding that covers the entire search space Ω. The crucial issue of AFSA is the 'visual scope' of each point $\mathbf{x}^i$. This represents a closed neighborhood of $\mathbf{x}^i$ with a radius equal to a positive quantity ν defined by

$$\nu = \delta \max_{j\in\{1,2,\ldots,n\}} (u_j - l_j)$$

where $\delta \in (0,1)$ is a positive visual parameter. This parameter may be reduced along the iterative process. Let $\mathbf{I}^i$ be the set of indices of the points inside the 'visual scope' of point $\mathbf{x}^i$, where $i \notin \mathbf{I}^i$ and $\mathbf{I}^i \subset \{1,2,\ldots,N\}$, and let np^i be the number of points in its 'visual scope'. Depending on the relative positions of the points in the population, three possible situations may occur:

- when $np^i = 0$, the 'visual scope' is empty, and the point $\mathbf{x}^i$, with no other points in its neighborhood, moves randomly looking for a better region;
- when the 'visual scope' is not crowded, the point $\mathbf{x}^i$ is able either to chase moving towards the best point inside the 'visual scope', or, if this best point does not improve the objective function value corresponding to $\mathbf{x}^i$, to swarm moving towards the central point of the 'visual scope';
- when the 'visual scope' is crowded, the point $\mathbf{x}^i$ has some difficulty in following any particular point, and searches for a better region by choosing randomly another point (from the 'visual scope') and moving towards it.

The condition that decides when the 'visual scope' of $\mathbf{x}^i$ is not crowded is

$$\frac{np^i}{N} \leq \theta, \tag{2}$$

where $\theta \in (0,1)$ is the crowd parameter. In this situation, the point $\mathbf{x}^i$ has the ability to swarm or to chase. The swarming behavior is characterized by a movement towards the central point inside the 'visual scope' of $\mathbf{x}^i$ defined by

$$\bar{\mathbf{x}} = \frac{\sum_{l \in \mathbf{I}^i} \mathbf{x}^l}{np^i}.$$

We refer the reader to [31,32,38,39] for details.

3 A New Simplified Binary Artificial Fish Swarm Algorithm

In the preliminary binary version of AFSA [3], each trial point is created from the current one by using the original concept of 'visual scope' of a current point. To identify the points inside the 'visual scope' of each current point, the Hamming distance is used. For points of equal bits length, this distance is the number of positions at which the corresponding bits are different. The computational requirement of this procedure grows rapidly with problem's dimension. Furthermore, in some cases the population stagnates and the algorithm converges to a non-optimal solution. To address these issues, the recent S-bAFSA was proposed with the following properties [4]:

- the concept of 'visual scope' of an individual point is discarded;
- the selection of each fish/point behavior does not depend on the number of points in the neighborhood of that point but rather on two target probability values;
- the swarming behavior is never performed since the central point may not depict the center of the distribution of solutions;
- uniform crossover is used in different fish behaviors, to create the trial points;
- a random heuristic drop_item procedure to make infeasible solutions to feasible ones, and an add_item operation, are combined to further improve the feasible solutions;
- drop_item and add_item are only used for a single linear capacity constraint (in the quadratic knapsack problem context);
- a simple heuristic search based on swap moves is implemented on a predefined number of points randomly selected from the population, aiming to obtain more accurate solutions;
- swap moves depend on the number of 0s in a point;
- the population is randomly reinitialized to diversify the search and avoid convergence to a non-optimal solution.

Nevertheless, in this study, S-bAFSA is further modified in order to increase efficiency and to enhance the quality of the solutions, for solving large 0–1 MKP of the form (1). The modifications are the following:

- an effect-based crossover is used instead of an uniform crossover;

- a random heuristic drop_item procedure is extended to handle m knapsack constraints in order to make infeasible solutions to feasible ones;
- a greedy-like heuristic add_item procedure preceded by drop_item is implemented to handle m constraints in order to further improve the feasible solutions;
- a simple local search with two steps is implemented. First, a flip-bit mutation is operated on a pre-defined number of points randomly selected from the population, with a pre-specified probability; second, at the end of the selection procedure, the best point is refined using a flip-bit mutation on a pre-defined number of positions.

Details of the proposed newS-bAFSA for solving the 0–1 multidimensional knapsack problem (1) are described below.

3.1 Initialization

The first step of newS-bAFSA is to design a suitable representation scheme of an individual point in a population for solving the 0–1 MKP. Since we consider the 0–1 knapsack problem, N current points, $\mathbf{x}^i, i = 1, \ldots, N$, each represented by a binary string of 0/1 bits of length n, are randomly generated [3,25]. We note that there are at most 2^n possible different solutions of binary strings of 0/1 bits of length n. For example, for $n = 12$, a current point $\mathbf{x}^i,\ i = 1, 2, \ldots, N$ randomly initialized at iteration $t = 1$ is shown in Fig. 1.

$\mathbf{x}^{i,1} =$

1	0	0	0	1	1	0	1	0	1	1	0

Fig. 1 Individual representation in newS-bAFSA

3.2 Generating Trial Points in newS-bAFSA

After initializing N current points, crossover and mutation are performed to create trial points in successive iterations based on the fish behaviors of random, chasing and searching. We introduce the probabilities $0 \leq \tau_1 \leq \tau_2 \leq 1$ in order to perform the movements of random, chasing and searching.

In newS-bAFSA, the fish behaviors that create the trial points are the following.

Random behavior: In random behavior, a fish with no other fish in its neighborhood to follow, moves randomly looking for food in another region. This behavior is implemented when an uniformly distributed random number $rand(0, 1)$ is less than or equal to τ_1. In this behavior the trial point $\mathbf{y}^i$ is created by randomly setting 0/1 bits of length n.

Chasing behavior: The chasing behavior is implemented when a fish, or a group of fish in the swarm, discover food and the others find the food dangling quickly after it. This behavior is implemented when $rand(0, 1) \geq \tau_2$ and it is related to the movement towards the best point found so far in the population, $\mathbf{x}^{\text{best}}$. Here, the trial point $\mathbf{y}^i$ is created using an effect-based crossover (see Algorithm 1) between $\mathbf{x}^i$ and $\mathbf{x}^{\text{best}}$.

Searching behavior: When fish discovers a region with more food, by vision or sense, it goes directly and quickly to that region. This is the searching behavior and is related to the movement towards a point $\mathbf{x}^{\text{rand}}$ where 'rand' is an index randomly chosen from the set $\{1, 2, \ldots, N\}$. This behavior is implemented in newS-bAFSA when $\tau_1 < rand(0, 1) < \tau_2$. An effect-based crossover between $\mathbf{x}^{\text{rand}}$ and $\mathbf{x}^i$ is performed to create the trial point $\mathbf{y}^k$.

Trial point corresponding to the best point: In newS-bAFSA, the three fish behaviors previously described are implemented to create $N - 1$ trial points; the best point $\mathbf{x}^{\text{best}}$ is treated separately. A 4 flip-bit mutation is performed on the point $\mathbf{x}^{\text{best}}$ to create the corresponding trial

point $\mathbf{y}$. In this operation four positions are randomly selected and the bits of the corresponding positions are changed from 0 to 1 or vice versa.

Assuming that fish move mostly together, it is natural to argue that the searching behavior is operated mostly since in the original AFSA this behavior occurs when the 'visual scope' is crowded. This argument has been transposed into the newS-bAFSA, since with a small value for τ_1 and a large value for τ_2, the likelihood that the searching behavior be implemented is higher than those of random or chasing behaviors.

3.3 The Effect-based Crossover

In the earlier work of bAFSA [3], both one point crossover and uniform crossover were implemented and found that the uniform crossover gives better results. Now, in newS-bAFSA, an *effect-based* crossover [7] is used in chasing and searching behaviors in order to create the trial points. In this crossover, each bit of the trial point is created by copying the corresponding bit from one or the other current point based on the *effect ratio*. To compute the effect ratio $ER_{\mathbf{u},\mathbf{x}^i}$ of $\mathbf{u}$ on the current point $\mathbf{x}^i$, where

- $\mathbf{u} = \mathbf{x}^{\text{best}}$ when chasing is performed;
- $\mathbf{u} = \mathbf{x}^{\text{rand}}$ when searching is performed;

we use

$$ER_{\mathbf{u},\mathbf{x}^i} = \frac{q(\mathbf{u})}{q(\mathbf{u}) + q(\mathbf{x}^i)}$$

where

$$q(\mathbf{x}^i) = exp\left[-(z(\mathbf{x}^{\text{best}}) - z(\mathbf{x}^i))/(z(\mathbf{x}^{\text{best}}) - z(\mathbf{x}^{\text{worst}}))\right]$$

and $\mathbf{x}^{\text{worst}}$ is the worst point of the population. The *effect-based* crossover to compute the trial point $\mathbf{y}^i$ is displayed in Algorithm 1.

Algorithm 1 Effect-based crossover

Require: current point $\mathbf{x}^i$, $\mathbf{u}$ and $ER_{\mathbf{u},\mathbf{x}^i}$
for $j = 1$ to n **do**
  if $rand(0,1) < ER_{\mathbf{u},\mathbf{x}^i}$ **then**
    $y_j^i = u_j$
  else
    $y_j^i = x_j$
  end if
end for
return trial point $\mathbf{y}^i$

3.4 Dealing with Knapsack Constraints

In binary represented population-based solution methods there exist different techniques for dealing the constraints. To obtain feasible solutions in newS-bAFSA, a general random heuristic procedure called drop_item to handle m knapsack constraints is used (see Algorithm 2). In this procedure one item is dropped from the knapsack (changing bit 1 to 0) if it does not satisfy all the knapsack constraints and it continues until the feasible solution is reached. A set $\mathbf{I} = \{I_1, I_2, \ldots, I_n\}$ is defined with n randomly generated indices. Then the drop_item is performed on $\mathbf{x}^i$ using the set $\mathbf{I}$ to make the point feasible. The advantage of this procedure is that dropping an item starts from any index and randomly continues selecting an index until the feasible solution is reached, aiming to obtain a promising solution.

After making the points feasible, a greedy-like heuristic procedure called add_item (Algorithm 3) that can handle m knapsack constraints is implemented to each feasible point aiming to improve

Algorithm 2 Drop_item algorithm used in newS-bAFSA

Require: Point $\mathbf{x}^i$ and the set $\mathbf{I} = \{I_1, I_2, \ldots, I_n\}$
Compute $\text{sum}_k = \sum_{j=1}^{n} a_{k,j} x_j^i$, for $k = 1, 2, \ldots, m$
Set flag := 1
for $k = 1$ to m **do**
  if $\text{sum}_k > b_k$ **then**
    Set flag := 0
    break
  end if
end for
if flag = 0 **then**
  for $j = 1$ to n **do**
    if $x_{I_j}^i = 1$ **then**
      Set flag1 := 1
      for $k = 1$ to m **do**
        if $\text{sum}_k - a_{k,I_j} > b_k$ **then**
          Set flag1 := 0
          break
        end if
      end for
      if flag1 = 0 **then**
        Set $x_{I_j}^i := 0$
        for $k = 1$ to m **do**
          Set $\text{sum}_k := \text{sum}_k - a_{k,I_j}$
        end for
      end if
    end if
  end for
end if
return Feasible point $\mathbf{x}^i$

the point and keep it feasible. When solving a single knapsack problem, this add_item utilizes only the information of the *pseudo-utility* ratios, δ_j, which are defined as the ratios of the objective function coefficients (c_j's) to the coefficients of the constraint (a_j's). The greater the ratio, the higher the chance that the corresponding variable will be equal to one in the solution [10]. In the generalization of this add_item heuristic for the 0–1 multidimensional knapsack problems, the *pseudo-utility* ratios of every item in every constraint are calculated, and only the lowest value for each item is considered (i.e., $\delta_j = \min\{(c_j b_k)/a_{k,j}\}$ $j = 1, \ldots, n, k = 1, \ldots, m$). Then δ_j are sorted in decreasing order and a set $\mathbf{J} = \{J_1, J_2, \ldots, J_n\}$ is defined with the indices of the δ_j in decreasing order. One item is added each time in the knapsack (changing bit 0 to 1) if it satisfies all the constraints following the sequence of indices in the set $\mathbf{J}$. This procedure is continued until the entire sequence of indices has been used.

The *pseudo-utility* ratios can also be used to make the solutions feasible. In this case the ratios are sorted in increasing order and one item is dropped from the knapsack, if with this item the solution violates any constraint. This procedure is continued until the feasible solution is reached.

3.5 Selection of a New Population

At each iteration, each trial point $\mathbf{y}^i$ competes with the current $\mathbf{x}^i$, in order to decide which one should become a member of the population in the next iteration. Hence, if $z(\mathbf{y}^i) \geq z(\mathbf{x}^i)$, then the trial point becomes a member of the population in the next iteration, otherwise the current point is preserved to the next iteration.

Algorithm 3 Add_item algorithm used in newS-bAFSA

Require: Feasible point $\mathbf{x}^i$ and set $\mathbf{J} = \{J_1, J_2, \ldots, J_n\}$
Compute $\text{sum}_k = \sum_{j=1}^{n} a_{k,j} x_j^i$, for $k = 1, 2, \ldots, m$
for $j = 1$ to n **do**
  if $x^i_{J_j} = 0$ **then**
    Set flag := 1
    for $k = 1$ to m **do**
      if $\text{sum}_k + a_{k,J_j} > b_k$ **then**
        Set flag := 0
        break
      end if
    end for
    if flag = 1 **then**
      Set $x^i_{J_j} := 1$
      for $k = 1$ to m **do**
        Set $\text{sum}_k := \text{sum}_k + a_{k,J_j}$
      end for
    end if
  end if
end for
return Improved feasible point $\mathbf{x}^i$

3.6 Reinitialization of the Population

When testing bAFSA [3], it was noticed that, in some cases, the points in a population converge to a non-optimal point. In a fish swarm context, this may be considered a region with food shortage for all the swarm, which may lead to the departure of part of the swarm looking for a better region. Considering that this situation may be occurring only in a seasonal way, we adopt a reinitialization of part of the population only at every R iterations. The parameter R gives the seasonal time period.

Hence in newS-bAFSA, to diversify the search and look for a promising region, a randomly reinitialization of 50% of the population, guaranteeing that the best solution found so far is maintained, is implemented. In practical terms, this technique has greatly improved the quality of the solutions.

3.7 Local Search

A local search is often important to improve a current set of solutions since further exploitation around a particular good region may provide high quality solutions. If a solution better than the current ones is found then it replaces one solution of the current set. In a fish swarm context, our local search may be interpreted by the swarm as a procedure that allows first of all a small percentage of the swarm to progress towards food and thereon confers to the best one further progress into a promising region.

In newS-bAFSA, the local search is based on a flip-bit mutation that is operated on N_{loc} points selected randomly from the population, where $N_{\text{loc}} = \tau_3 N$ with $\tau_3 \in (0, 1)$. This flip-bit mutation that operates on a point changes the value of a 0 bit to 1 and vice versa according to a probability p_m. After the flip-bit operation, the new points are made feasible by using the random drop_item procedure and thereon the add_item. Then they become members of the population if they improve the objective function value with respect to the corresponding current points. This flip-bit operation is repeated L times in order to find good solutions. At the end, the best point of the population is identified and a flip-bit mutation is operated on N_{ref}, with $N_{\text{ref}} = \tau_3 n$, randomly selected positions of the point. Each time a new point is created, the random drop_item and thereon the add_item are implemented to make the point feasible. Then this new point will replace the best point if it improves the objective function value with respect to the current best point.

3.8 Termination Conditions

The proposed newS-bAFSA terminates when the known optimal solution is reached within a tolerance $\epsilon > 0$, or a maximum number of iterations, $T_{\max}$, is exceeded, i.e., when

$$t > T_{\max} \text{ or } |z_{\text{best}} - z_{\text{opt}}| \leq \epsilon \tag{3}$$

holds, where z_{best} is the best objective function value attained at iteration t and z_{opt} is the known optimal value available in the literature. However, if the optimal value of the given problem is not known, the algorithm may use another condition, for example, one based on the total number of function evaluations or the computational time since the start of the algorithm.

3.9 The Algorithm

The pseudocode of the herein proposed newS-bAFSA for solving the 0–1 MKP (1) is shown in Algorithm 4.

Algorithm 4 newS-bAFSA

Require: $T_{\max}$ and z_{opt} and other values of parameters
Set $t := 1$. Initialize population $\mathbf{x}^i, i = 1, 2, \ldots, N$
Perform random drop_item and add_item, evaluate the population and identify $\mathbf{x}^{\text{best}}$ and z_{best}
while 'termination conditions are not met' **do**
  if MOD$(t, R) = 0$ **then**
    Reinitialize 50% of the population (keep $\mathbf{x}^{\text{best}}$)
    Perform random drop_item and add_item, evaluate population and identify $\mathbf{x}^{\text{best}}$ and z_{best}
  end if
  for $i = 1$ to N **do**
    if $i =$ best **then**
      Perform 4 flip-bit mutation to create trial point $\mathbf{y}^i$
    else
      if $rand(0, 1) \leq \tau_1$ **then**
        Perform random behavior to create trial point $\mathbf{y}^i$
      else if $rand(0, 1) \geq \tau_2$ **then**
        Perform chasing behavior to create trial point $\mathbf{y}^i$
      else
        Perform searching behavior to create trial point $\mathbf{y}^i$
      end if
    end if
  end for
  Perform random drop_item and add_item to get $\mathbf{y}^i, i = 1, 2, \ldots, N$ and evaluate them
  Select the population of next iteration $\mathbf{x}^i, i = 1, 2, \ldots, N$
  Perform local search (identifying $\mathbf{x}^{\text{best}}$ and z_{best})
  Set $t := t + 1$
end while

4 Experimental Results

We code newS-bAFSA in C and compile with Microsoft Visual Studio 10.0 compiler in a PC having 2.5 GHz Intel Core 2 Duo processor and 4 GB RAM. We consider large-sized benchmark 0–1 MKP test instances (described in [10]) from OR-library[1]. These are instances with 5, 10 and 30 constraints and 100, 250 and 500 variables. The values of the tightness ratio α for resource capacities b_k ($b_k = \alpha \sum_{j=1}^{n} a_{kj}$, $k = 1, 2, \ldots, m$) are 0.25, 0.50 and 0.75. Ten instances for each $n \times m \times \alpha$ combination give a total of 270 instances.

[1] `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`

At first and to check the effectiveness of the local search, we compare two variants of the newS-bAFSA, one with the local search (LS) and the other without it. After several experiments, we set $N = n$, $T_{\max} = 2000$, $R = 100$, $L = 30$, $\tau_1 = 0.1$, $\tau_2 = 0.9$, $\tau_3 = 0.1$, $p_m = 0.1$ and $\epsilon = 10^{-4}$. We use the best solution reported in OR-library as a termination condition with error tolerance along with $T_{\max}$. Thirty independent runs were carried out for each instances using each variant. We compare the performance of the obtained results measured by the percentage (%) gap between the best objective function value and the optimal value of the LP (linear programming) relaxation, i.e.,

$$\text{Gap } \% = \frac{\text{optimal LP value} - \text{best objective value}}{\text{optimal LP value}} \times 100.$$

The comparison of the two variants using box plots is shown in Figures 2 – 4. In a box plot, the

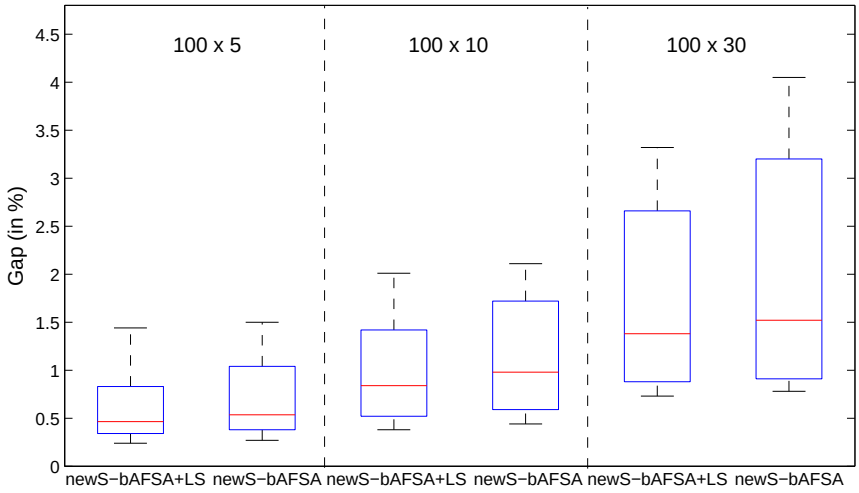

Fig. 2 Box plots of percentage gap for 100×5, 100×10 and 100×30

box contains 50% of the data, from lower quartile (box bottom line) to upper quartile (box top line), where the median is represented by the solid line inside the box. The lowest datum (lower whisker) is within $1.5IQR$ (inter quartile range) of the lower quartile, and the highest datum (higher whisker) is within $1.5IQR$ of the upper quartile. Any data not included between the whiskers is considered an outlier. From the figures we may conclude that the variant newS-bAFSA with LS improves over the other without LS, when solving the 270 0–1 MKP test instances.

Now, newS-bAFSA with LS is compared with other population-based heuristic methods available in the literature: genetic algorithm, GA [10], discrete electromagnetism-like mechanism, DEM [7] and set-based particle swarm optimization, SBPSO [24]. See Table 1, where the experimental results of the other methods were taken from the corresponding literature.

GA [10] is a heuristic based solution method for the 0–1 multidimensional knapsack problems. It uses $N = 100$ points of binary string in a population. The trial points are created using the uniform crossover and mutation after selecting parents using binary tournament selection procedure. The knapsack constraints are handled using drop-add repair operator based on pseudo-utility ratios. The algorithm terminates when 10^6 non-duplicate children are created. GA was tested on the 270 large 0–1 MKP instances. Just a single run was performed and the percentage gap, the computational time (in seconds) to find the best solution at first time were reported.

DEM [7] utilizes the operators of genetic algorithm in order to perform the movements of points (particles) instead of standard movement of the electromagnetism-like mechanism. It generates

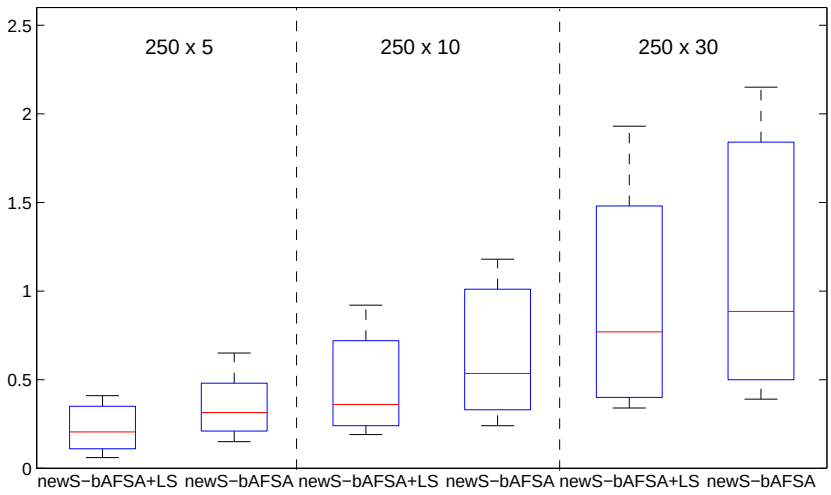

Fig. 3 Box plots of percentage gap for 250×5, 250×10 and 250×30

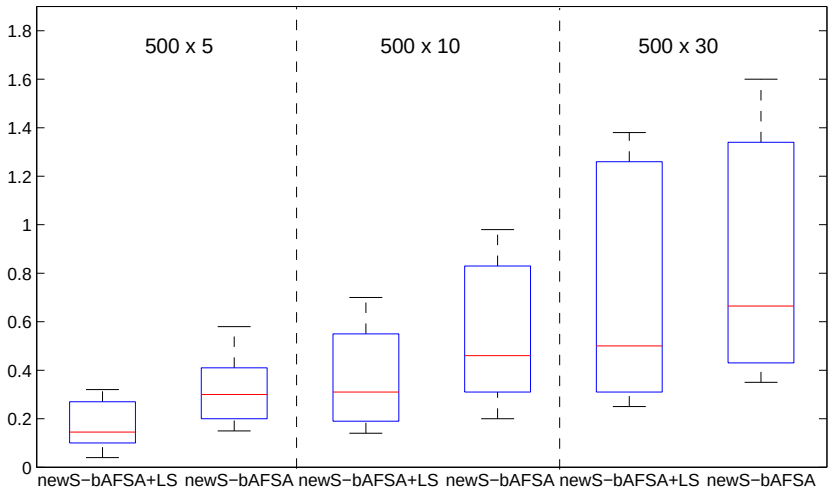

Fig. 4 Box plots of percentage gap for 500×5, 500×10 and 500×30

binary string to create $N = n$ points in a population. In order to create trial points, DEM uses max min effects-based uniform crossover and mutation after calculating the force of each point. An annihilation/creation is also used to overcome the stagnation of points in search space. It also uses the procedure drop-add repair operator described in [10] to make the solutions feasible. A local search described in [23] is also implemented. Hence the variant of DEM is referred as AMMDEM+LS [7]. The algorithm terminates when the maximum number of objective function evaluations is reached 10^4. AMMDEM+LS was also tested with the 270 large 0–1 MKP instances. Twenty five runs were carried out and the percentage gap, the computational time (in seconds) to find the best solution at first time were reported.

SBPSO [24] is a variant of standard PSO that can solve discrete optimization problems. It defines a point's (particle's) position and velocity as mathematical sets. SBPSO uses $N = 50$

points that are randomly initialized to form a population. The initial velocities are initialized as empty sets. The velocities and positions are updated by using sets of operation pairs and sets of elements from an universe set, respectively. In order to handle constraints SBPSO penalizes an infeasible solution by setting the objective function value to $-\infty$ (for maximization case). The algorithm stops if the obtained best objective function value is equal to the LP relaxed bound, the obtained best objective function value is not improved for 2500 iterations or the number of iterations exceeds 5000. SBPSO solved the 270 large 0–1 MKP instances, thirty runs were performed and the percentage gap was reported. Based on different topologies the global best set-based particle swarm optimization, GBSBPSO is reported as the best one among others [24].

Table 1 Comparative results of different algorithms

Prob. set		GA		AMMDEM+LS		GBSBPSO		newS-bAFSA	
m	n	Gap	T_a	Gap	AT_a	Gap	AT_a	Gap	AT_a^*
5	100	0.59	20.0	0.58	13.0	1.11	–	0.59	14.9
	250	0.14	174.4	0.14	25.0	1.86	–	0.22	127.4
	500	0.05	70.5	0.05	95.0	2.66	–	0.17	696.6
10	100	0.94	314.4	0.94	19.0	1.14	–	1.00	19.5
	250	0.30	276.8	0.28	31.0	1.53	–	0.46	164.6
	500	0.14	734.1	0.12	155.0	1.86	–	0.35	860.7
30	100	1.69	128.5	1.68	14.0	1.50	–	1.73	44.2
	250	0.68	847.9	0.65	57.0	1.86	–	0.90	328.8
	500	0.35	1384.0	0.34	252.0	1.98	–	0.70	1487.3
Average		0.54	438.9	0.53	73.4	1.72	–	0.68	416.0

– Not available, * average of total execution time

In Table 1, 'T_a' represents the average computational time (in seconds) of a problem set, 'AT_a' represents the average computational time (in seconds) of a problem set among 30 runs. From the table we may observe that AMMDEM+LS outperforms the other methods in comparison with respect to the both performance criteria. Based on criterion 'Gap', newS-bAFSA gives better performance than GBSPSO, except on problem set 100×30. Overall newS-bAFSA gives a percentage gap of 0.68% and requires 416.0 seconds of computational time.

Finally, we compare the proposed newS-bAFSA with a binary ant system (BAS) presented in [23], for the problem sets 100×5 and 100×10. BAS uses $N = 30$ ants/points in a population that are initialized randomly of binary string. In order to make the solutions feasible, BAS uses the procedure drop-add repair operator described in [10], and also implements a local search. The algorithm terminates when the number of iterations exceeds $T_{\max} = 3000$ or when the best available solution is reached. Thirty independent runs were carried out for each instance. We note that our newS-bAFSA algorithm was run with $N = n$ and $T_{\max} = 2000$ since these values provide the best performance among others. Our experiments also showed that a larger value for $T_{\max}$ has no significant effect on the quality of the solutions.

The comparative results are shown in Tables 2 and 3. The results of BAS are taken from the corresponding literature. The performance criteria are 'z_{best}', the obtained best objective function value and 'AT_a', the average computational time (in seconds) of a problem set among 30 runs.

From Tables 2 and 3, it is shown that BAS gives better performance based on the performance criterion z_{best}, while newS-bAFSA gives in general better performance based on AT_a for both sets of problems, although we note that the machine used for both algorithms are different. When comparing the best objective function value, BAS wins on 10 instances, newS-bAFSA wins on one and 49 values are ties. Thus, from the 82% of ties, BAS has smaller average computational time on 43% of the instances and newS-bAFSA has smaller AT_a on 57% of the instances.

Based on the numerical experiments carried out, we may conclude that the new proposed simplified binary version of the artificial fish swarm algorithm is rather effective when solving the 0–1 multidimensional knapsack problems.

Table 2 Comparison of BAS and newS-bAFSA for 100×5 set (bold values show the best obtained results)

		BAS		newS-bAFSA	
Inst.	α	z_{best}	AT_a	z_{best}	AT_a
1		**24381**	**1.52**	**24381**	18.97
2		**24274**	22.63	**24274**	**18.18**
3		**23551**	82.40	**23551**	**19.85**
4		**23534**	141.45	**23534**	**19.91**
5	0.25	**23991**	**12.87**	**23991**	19.65
6		**24613**	**3.20**	**24613**	19.43
7		**25591**	**0.25**	**25591**	10.30
8		**23410**	**0.95**	**23410**	18.29
9		**24216**	135.60	**24216**	**18.34**
10		**24411**	**11.50**	**24411**	19.45
11		**42757**	38.96	**42757**	**16.83**
12		**42545**	203.08	42536	17.38
13		**41968**	173.23	41967	17.25
14		**45090**	74.74	45071	17.15
15	0.50	**42218**	**4.18**	**42218**	12.84
16		**42927**	**1.63**	**42927**	16.61
17		**42009**	**1.28**	**42009**	10.84
18		**45020**	**14.26**	**45020**	16.95
19		**43441**	60.58	**43441**	**16.90**
20		**44554**	**11.22**	**44554**	16.51
21		**59822**	**6.61**	**59822**	9.73
22		**62081**	139.41	**62081**	**11.25**
23		**59802**	113.73	**59802**	**10.79**
24		**60479**	69.74	**60479**	**8.44**
25	0.75	**61091**	59.42	**61091**	**11.35**
26		**58959**	**3.77**	**58959**	10.23
27		**61538**	35.20	**61538**	**11.87**
28		**61520**	52.15	**61520**	**9.79**
29		**59453**	**2.03**	**59453**	9.38
30		**59965**	50.87	**59965**	**11.04**
Average			50.95		**14.85**

5 Conclusions

In this paper, a new simplified binary version of the artificial fish swarm algorithm for solving large 0–1 MKP has been presented. In this method, denoted by newS-bAFSA, a point in the population is represented by a binary string of 0/1 bits. The movement of a point is characterized by random, chasing or searching behavior according to two user defined probability values. To create the trial points, chasing and searching behaviors are carried out by means of an effect-based crossover. However, a 4 flip-bit mutation is performed on the current best point to create the corresponding trial point. A random heuristic drop_item procedure is implemented to make the points feasible. A simple greedy-like algorithm add_item is also used to improve the quality of the solutions. To enhance the search for an optimal solution, a flip-bit mutation based local search is performed on some randomly selected points from the population and thereon on some randomly chosen positions of the best point. A cyclic reinitialization of 50% of the population is also introduced to improve the quality of the solutions. A final remark concerned with the design of newS-bAFSA. The most time consuming procedures of the original binary AFSA are the computation of the Hamming distances and the comparison between those distances and the 'visual scope'. Both procedures are required to select the behavior for each fish move in the swarm. Although newS-bAFSA relies on other procedures to choose better and better solutions, the selection of fish behaviors, their implementations, and the local search have indeed been the focus of this simplification process. They were replaced by simple and easy to implement randomization processes.

Table 3 Comparison of BAS and newS-bAFSA for 100 × 10 set (bold values show the best obtained results)

		BAS		newS-bAFSA	
Inst.	α	z_{best}	AT_a	z_{best}	AT_a
1		**23064**	28.89	**23064**	**25.24**
2		**22801**	38.85	**22801**	**17.49**
3		**22131**	26.61	22081	26.88
4		**22772**	**5.90**	**22772**	25.18
5	0.25	**22751**	130.27	**22751**	**25.25**
6		**22777**	247.77	22725	27.02
7		**21875**	31.81	21841	27.02
8		**22635**	**14.63**	**22635**	25.32
9		**22511**	91.87	22423	26.93
10		**22702**	**0.87**	**22702**	12.11
11		**41395**	38.53	**41395**	**23.67**
12		**42344**	163.70	**42344**	**15.18**
13		**42401**	57.83	**42401**	**23.55**
14		**45624**	186.84	45475	24.22
15	0.50	**41884**	359.18	**41884**	**21.19**
16		**42995**	18.62	**42995**	**14.06**
17		**43574**	67.44	43559	23.70
18		**42970**	**18.62**	**42970**	24.04
19		**42212**	**0.58**	**42212**	21.90
20		**41207**	218.88	41123	24.39
21		**57375**	25.59	**57375**	**2.17**
22		**58978**	83.20	**58978**	**17.81**
23		**58391**	34.22	**58391**	**15.82**
24		**61966**	141.42	**61966**	**16.51**
25	0.75	**60803**	**3.24**	**60803**	3.65
26		**61437**	227.48	**61437**	**18.71**
27		56353	133.99	**56377**	18.22
28		**59391**	**5.26**	**59391**	18.13
29		**60205**	194.22	**60205**	**13.85**
30		**60633**	10.18	**60633**	**5.06**
Average			86.88		**19.48**

A comparison of newS-bAFSA with other solution methods available in the literature has been carried out with 270 benchmark test instances. It is found that the proposed method is rather competitive when solving large 0–1 multidimensional knapsack problems. Future work will consider using newS-bAFSA to solve general multidimensional knapsack problems effectively. Other NP-hard challenging combinatorial optimization problems will be also addressed in the future.

Acknowledgments

The authors wish to thank three anonymous referees for their comments and valuable suggestions to improve the paper.

The first author acknowledges Ciência 2007 of FCT (Foundation for Science and Technology) Portugal for the fellowship grant C2007-UMINHO-ALGORITMI-04. Financial support from FEDER COMPETE (Operational Programme Thematic Factors of Competitiveness) and FCT under project FCOMP-01-0124-FEDER-022674 is also acknowledged.

References

1. Akçay Y., Li H., Xu S.H.: Greedy algorithm for the general multidimensional knapsack problem. Annals of Operations Research **150**, 17–29 (2007)
2. Al-Shihabi S., Ólafsson S.: A hybrid of nested partition, binary ant system, and linear programming for the multidimensional knapsack problem. Computers & Operations Research **37**, 247–255 (2010)

3. Azad M.A.K., Rocha A.M.A.C., Fernandes E.M.G.P.: Solving multidimensional 0-1 knapsack problem with an artificial fish swarm algorithm. In: Murgante, B. et al. (eds.), Computational Science and Its Applications, ICCSA 2012, Part III, LNCS, Vol. 7335. pp. 72–86, Springer-Verlag, Heidelberg (2012)
4. Azad M.A.K., Rocha A.M.A.C., Fernandes E.M.G.P.: A simplified binary artificial fish swarm algorithm for 0–1 quadratic knapsack problems, Journal of Computational and Applied Mathematics **259**, 897–904 (2014)
5. Balev S., Yanev N., Fréville A., Andonov R.: A dynamic programming based reduction procedure for the multidimensional 0–1 knapsack problem. European Journal of Operational Research **166**, 63–76 (2008)
6. Bansal J.C., Deep K.: A modified binary particle swarm optimization for knapsack problems. Applied Mathematics and Computation **218**(22), 11042–11061 (2012)
7. Bonyadi M.R., Li X.: A new discrete electromagnetism-based meta-heuristic for solving the multidimensional knapsack problem using genetic operators. Operational Research - An International Journal **12**, 229–252 (2012)
8. Boyer V., Elkihel M., Baz D.E.: Heuristics for the 0–1 multidimensional knapsack problem. European Journal of Operational Research **199**, 658–664 (2009)
9. Cabot A.V.: An enumeration algorithm for knapsack problems. Operations Research **18**, 306–311 (1970)
10. Chu P.C., Beasley J.E.: A genetic algorithm for the multidimensional knapsack problem. Journal of Heuristics **4**, 63–86 (1998)
11. Costa M.F.P., Rocha A.M.A.C., Fernandes, E.M.G.P.: An artificial fish swarm algorithm based hyperbolic augmented Lagrangian method, Journal of Computational and Applied Mathematics **259**, 868–876 (2014)
12. Djannaty F., Doostdar S.: A hybrid genetic algorithm for the multidimensional knapsack problem. International Journal of Contemporary Mathematical Sciences **3**(9), 443–456 (2008)
13. Drexl A.: A simulated annealing approach to the multiconstraint zero–one knapsack problem. Computing **40**, 1–8 (1988)
14. Fontanari J.F.: A statistical analysis of the knapsack problem. Journal of Physics A: Mathematical and General **28**, 4751–4759 (1995)
15. Fréville A.: The multidimensional 0–1 knapsack problem: An overview. European Journal of Operational Research **155**, 1–21 (2004)
16. Fréville A., Plateau G.: The 0–1 bidimensional knapsack problem: Towards an efficient high-level primitive tool. Journal of Heuristics **2**, 147–167 (1996)
17. Gavish B., Pirkul H.: Efficient algorithms for solving multiconstraint zero–one knapsack problems to optimality. Mathematical Programming **31**, 78–105 (1985)
18. Hanafi S., Fréville A.: An efficient tabu search approach for the 0–1 multidimensional knapsack problem. European Journal of Operational Research **106**, 659–675 (1998)
19. Hill R.R., Cho Y.K., Moore J.T.: Problem reduction heuristic for the 0–1 multidimensional knapsack problem. Computers & Operations Research **39**, 19–26 (2012)
20. Jiang M., Mastorakis N., Yuan D., Lagunas M.A.: Image segmentation with improved artificial fish swarm algorithm. in: Mastorakis N. et al. (eds.), ECC 2008, LNEE, Vol. 28. pp. 133–138, Springer-Verlag, Heidelberg (2009)
21. Jiang M., Wang Y., Pfletschinger S., Lagunas M.A., Yuan D.: Optimal multiuser detection with artificial fish swarm algorithm. In: Huang D.S. et al. (eds.), ICIC 2007, CCIS, Vol. 2. pp. 1084–1093, Springer-Verlag, Heidelberg (2007)
22. Ke L., Feng Z., Ren Z., Wei X.: An ant colony optimization approach for the multidimensional knapsack problem. Journal of Heuristics **16**, 65–83 (2010)
23. Kong M., Tian P., Kao Y.: A new ant colony optimization algorithm for the multidimensional knapsack problem. Computers & Operations Research **35**, 2672–2683 (2008)
24. Langeveld J., Engelbrecht A.P.: Set-based particle swarm optimization applied to the multidimensional knapsack problem. Swarm Intelligence **6** 297–342 (2012)
25. Michalewicz Z.: Genetic Algorithms+Data Structures=Evolution Programs. Springer, Berlin, (1996)
26. Neshat M., Sepidnam G., Sargolzaei M., Toosi A.N.: Artificial fish swarm algorithm: a survey of the state-of-the-art, hybridization, combinatorial and indicative applications. Artificial Intelligence Review (2012). DOI:10.1007/s10462-012-9342-2
27. Petersen C.C.: Computational experience with variants of the Balas algorithm applied to the selection of R&D projects. Management Science **13**(9), 736–750 (1967)
28. Pirkul H.: A heuristic solution procedure for the multiconstraint zero-one knapsack problem. Naval Research Logistics **34** 161–172 (1987)
29. Pisinger D.: Algorithms for knapsack problems. Ph.D. thesis, Department of Computer Science, University of Copenhagen, Denmark, (1995)
`http://www.diku.dk/hjemmesider/ansatte/pisinger/`
30. Puchinger J., Raidl G.R., Pferschy U.: The multidimensional knapsack problem: structure and algorithms. INFORMS Journal of Computing **22**(2), 250–265 (2010)
31. Rocha A.M.A.C, Fernandes E.M.G.P., Martins T.F.M.C.: Novel fish swarm heuristics for bound constrained global optimization problems. In: Murgante B. et al. (eds.), Computational Science and Its Applications, ICCSA 2011, Part III, LNCS, Vol. 6784. pp. 185–199 Springer-Verlag, Heidelberg (2011)
32. Rocha A.M.A.C, Martins T.F.M.C, Fernandes E.M.G.P.: An augmented Lagrangian fish swarm based method for global optimization. Journal of Computational and Applied Mathematics **235**, 4611–4620 (2011)
33. Sakawa M., Kato K.: Genetic algorithms with double strings for 0–1 programming problems. European Journal of Operational Research **144**, 581–597 (2003)
34. Schilling K.E.: The growth of m–constraint random knapsacks. European Journal of Operational Research **46**, 109–112 (1990)

35. Shih W.: A branch and bound method for the multiconstraint zero–one knapsack problem. Journal of the Operational Research Society **30**, 369–378 (1979)
36. Soyster A.L., Lev B., Slivka W.: Zero–one programming with many variables and few constraints. European Journal of Operational Research **2**, 195–201 (1978)
37. Vasquez M., Vimont Y.: Improved results on the 0–1 multidimensional knapsack problem. European Journal of Operational Research **165**, 70–81 (2005)
38. Wang C.R., Zhou C.-L., Ma J.-W.: An improved artificial fish swarm algorithm and its application in feed-forward neural networks. In: Proceedings of the Fourth International Conference on Machine Learning and Cybernetics. pp. 2890–2894 (2005)
39. Wang X., Gao N., Cai S., Huang M.: An artificial fish swarm algorithm based and abc supported QoS unicast routing scheme in NGI. In: Min G. et al. (eds.), ISPA 2006, LNCS, Vol. 4331. pp. 205–214, Springer-Verlag, Heidelberg (2006)
40. Weingartner H.M., Ness D.N.: Methods for the solution of the multidimensional 0/1 knapsack problem. Operations Research **15**, 83–103 (1967)