Oper Res Int J
DOI 10.1007/s12351-010-0084-0

ORIGINAL PAPER

# A new discrete electromagnetism-based meta-heuristic for solving the multidimensional knapsack problem using genetic operators

**Mohammad Reza Bonyadi · Xiaodong Li**

Received: 6 February 2010 / Revised: 10 July 2010 / Accepted: 12 July 2010


**Abstract** The Standard Electromagnetism-like Mechanism (*SEM*) is one of the swarm-based optimization methods which is examined in this paper. The *SEM* works based on the charges in electrons and hence its operators have been especially designed for continuous space problems. Although the *SEM* was successfully applied to the standard optimization problems, it was not that notable when it came to tackling discrete space problems. This shortcoming was obvious when the *SEM* was applied to some standard discrete problems such as Travelling Salesman Problem, Nurse Scheduling Problem, etc. In this paper, a modified *SEM* called Discrete Electromagnetism-like Mechanism is proposed which utilizes Genetic Algorithm (*GA*) operators to work in discrete spaces. In fact, the vector calculations (which are at the heart of the *SEM*) in the *SEM* are replaced by specific types of GA operators to determine the effects that particles have on one another. Also, a new operator based on the principles of quantum mechanics is proposed which further improves the performance of the method. In our experiments, the proposed algorithm is applied to a well-studied discrete space problem called Multidimensional Knapsack Problem (*MKP*). All tests are done on standard problems of the *MKP* and the results are reported and compared with several stochastic population-based optimization methods. Experiments showed that the proposed algorithm not only found comparable (and even better in some cases) solutions for the standard problems of the *MKP*, but also took much less computational time (75% improvement in average in comparison to other methods).

M. R. Bonyadi (✉)
Shahid Beheshti University, Tehran, Iran
e-mail: m_bonyadi@std.sbu.ac.ir; vardiar@gmail.com; vardiar@yahoo.com

X. Li
RMIT University, Melbourne, VIC, Australia
e-mail: xiaodong.li@rmit.edu.au

Published online: 29 July 2010

**Keywords** Swarm intelligence · Population-based optimization · Electromagnetism-Like meta-heuristic · Multidimensional knapsack problem

## 1 Introduction

The Standard Electromagnetism-like Mechanism (*SEM*) is a heuristic introduced by Birbil and Fang (2003) and successfully applied to standard optimization problems. Also, the *SEM* was applied to the NP-Hard problems such as routing problems (Wu et al. 2006; Yurtkuran and Emel 2010, Scheduling Problems (Chang et al. 2009; Debels and Vanhoucke 2004, 2006; Maenhout and Vanhoucke 2007; Naderi et al. 2010), etc. and the results were promising. The main idea of the *SEM* is based on the attraction-repulsion mechanism of electromagnetism theory (Coulomb's law). This is similar to the charge of particles in elementary electromagnetism. In this approach, the *charge* of each point relates to the objective value (Birbil and Fang 2003). Although the results of applying the *SEM* to standard continuous space problems was highly satisfactory (Birbil and Fang 2003), these results for discrete space problems were not good enough and not comparable with other Meta heuristics such as Particle Swarm Optimization and Ant Colony Optimization for solving these kinds of problems. The main reason that the *SEM* cannot work properly in discrete space problems is that its operators (force calculation and movement) are not compatible with this kind of spaces.

One of the well-studied NP-Hard problems is the Multidimensional Knapsack Problem (MKP). The *MKP* requires finding a subset of a set of items such that the total profit is maximized while the resource constraints are not violated. Recently, the *MKP* has attracted a lot of attention because of its wide applications in industries and economies. It is one of the well-studied discrete programming problems which can formulate many practical problems (Martello and Toth 1990). Capital budgeting problem, allocating processors and databases in a distributed computer system, project selection and cargo loading, and cutting stock are some problems that can be modeled by the *MKP*. There are two general approaches for solving the *MKP*: the exact algorithms (Gilmore and Gomory 1966; Shih 1979) that are used for solving small to moderate-size instances of the *MKP* and the approximation algorithms that are used to approximately solve difficult optimization problems. The meta-heuristic algorithms are in second group (approximation methods).

In this paper, a new version of *SEM* called Discrete Electromagnetism-like Mechanism (*DEM*) is proposed which is highly compatible with discrete space problems. In fact, operators of *SEM* such as force calculation and moving particles are modified such that they can work properly in discrete spaces. Also, a new operator called Annihilation/Creation ($A/A^{\dagger}$) is proposed which is inspired by a phenomenon in quantum mechanics and can enormously improve the performance of the algorithm. The proposed algorithm is applied to the standard test benches of the *MKP* and its results are compared with other new methods.

The remainder of the paper is organized as follows; in Sect. 2, a brief background on *SEM* and some related works for solving the *MKP* are given. Section 3 consists of two main parts. The first part specifies the general case of the *DEM* and presents

its operators. In the second part of Sect. 3 the proposed *DEM* for solving the *MKP* is introduced. Afterwards, in Sect. 4, we first set the parameters of the *DEM* and then compare the simulation results with other state-of-the-art methods.

## 2 Background

### 2.1 Standard electromagnetism-like mechanism

The *SEM* method utilizes an attraction-repulsion mechanism to move the sample points towards the optimality. The main focus of the first introduction of this heuristic was on the problems with bounded variables in the form of Eq. (1).

$$\text{Max } f(y) \quad \text{such that} \quad y \in [\boldsymbol{l}, \boldsymbol{u}] \tag{1}$$

$$[\boldsymbol{l}, \boldsymbol{u}] = \{y \text{ in } \mathbb{R}^n | l_k \leq y_k \leq u_k, k = 1, \ldots n\} \tag{2}$$

The *SEM* consists of three main phases. These are *charge calculation, calculation of the total force* on each particle, and *movement* along the direction of the force. For evaluation, the *SEM* uses the charges of the particles according to their objective values. However, in this mechanism the charge of each particle is not constant and it changes from generation to generation. The charge for each particle is calculated according to the following formula:

$$q^i = \exp\left(-n \frac{f(y^i) - f\left(y^{\text{best}}\right)}{\sum_{k=1}^{\text{mp}} \left(f(y^k) - f(y^{\text{best}})\right)}\right) \tag{3}$$

In Eq. (3), mp is the number of particles and *f(y)* is the objective value of particle $y$. In a maximization problem, the force of particle $i$ is calculated as follows:

$$\boldsymbol{F}^i = \sum_{j \neq i}^{\text{mp}} \left\{ \begin{array}{ll} (y^j - y^i) \frac{q^i q^j}{\|y^i - y^j\|^2} & f(y^j) > f(y^i) \\ (y^i - y^j) \frac{q^i q^j}{\|y^i - y^j\|^2} & f(y^j) \leq f(y^i) \end{array} \right\}, \quad \forall i \tag{4}$$

In fact, the forces are the effects on a particle from other ones. After the total force vector for the $i$th particle is evaluated, the particle is moved in the direction of the force with the step length of $\lambda$ which is selected randomly between 0 and 1. The following formula is used for the movement of particles:

$$y_k^i = \left\{ \begin{array}{ll} y_k^i + \lambda \frac{F_k^i}{\|F^i\|}(u_k - y_k^i) & \text{if } F_k^i > 0 \\ y_k^i + \lambda \frac{F_k^i}{\|F^i\|}(y_k^i - l_k) & \text{if } F_k^i \leq 0 \end{array} \right. \tag{5}$$

where $y_k^i$is the $k$th element of $i$th particle, $F_k^i$ is the calculated force on the $k$th element of $i$th particle and, $\|.\|$ is the norm operator. Formula (5) shows the movement equation. Note that the best particle is not moved and is carried to the next generation.

Figure 1 represents an example. As it is stated in the figure, particles 1, 2 and 3 have the objective values equal to 10, 15 and 20, respectively. Here, the aim is to calculate the force on particle 2. The problem is maximization and particle 1 is the

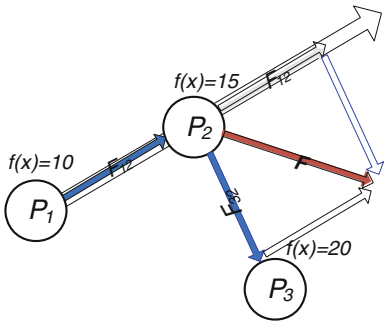


**Fig. 1** $F_{12}$ is the force from particle 1 to particle 2 (repulsion) and $F_{32}$ is the force from particle 3 to particle 2 (attraction), F is the resultant force vector. *Algorithm* 1 shows the procedure of *SEM*

**Algorithm 1** Solve a problem using *SEM*

*1) Initialize particles*
*2) While stopping criteria not satisfied*
*a. Calculate objective values for all particles*
*b. Calculate charges according to Eq.* 3
*c. Calculate forces according to Eq.* 4
*d. Move particles according to Eq.* 5
*3) Return the best particle as the best found solution*

worst (lowest objective value). So this particle shows a repulsion force on particle 2. Particle 3 is better than particle 2 and it represents an attraction force on particle 2 and finally the force F is calculated.

## 2.2 Multidimensional knapsack problem and related works

The Multidimensional Knapsack problem (MKP) is one of the well-studied discrete programming problems which can formulate many practical problems (Martello and Toth 1990). The problem is as follows: there are $n$ objects, each of them has a price $p_j$ and $m$ knapsacks, each of which has a capacity of $b_i$. Each object $j$ occupies $w_{ij}$ unit space of each knapsack $m_j$. The goal is to pick a subset of the objects such that no knapsack is overfilled and also the sum of the prices of selected objects to maximize. Formally, *MKP* can be formulated as

$$h(x_1, \ldots, x_n) = \sum_{j=1}^{n} p_j x_j \tag{6}$$

$$\sum_{j=1}^{n} w_{ij} x_j \leq b_i, i = 1, \ldots, m \tag{7}$$

$$x_j \in \{0, 1\}, j = 1, \ldots, n \tag{8}$$

The aim is to find the maximum value of $h$ such that Eq. (7) and (8) are satisfied. Condition (7) is known as the knapsack constraint and causes to call the problem as $M$-dimensional (Multidimensional) Knapsack Problem (MKP). Equation (8) describes the state of the variable $x$ and makes the problem as the Zero/One

Multidimensional Knapsack Problem. It is well known that the MKP is strongly NP-Hard because there is no polynomial approximation algorithm to solve it (Garey and Johnson 1979).

The *MKP* has recently attracted a lot of attention by researchers because of its wide applications. One of the earliest approaches for solving the *MKP* by genetic algorithm was presented by Khuri et al. (1994). They used the string 0/1 representation for their chromosomes and the one point crossover for the crossover operator. They introduced a fitness function that utilized a graded penalty term. Also, another method was proposed by Cotta and Troya (1998) which used a heuristic method in combination with GA. Another method for the problem was introduced in Alves and Almeida (2008). The paper presented a new multi-objective genetic algorithm based on the Tchebycheff scalarizing function, which aims to generate a good approximation of the non-dominated solution set of the multi-objective problem. Another well-known method which was based on GA was presented by Chu and Beasley (1998) and introduced very promising solutions for some standard test benches.

Recently, a number of ant-based algorithms have been employed for the *MKP* (Alaya et al. 2004; Fidanova 2002; Ke et al. 2010; Kong et al. 2008; Leguizamon and Michalewicz 1999; Rafael and Nikitas 2003). Also, the *PSO* (Kong and Tian 2007), Tabu search (Hanafi and Fréville 1998), scatter search (Hanafi and Wilbaut 2008), and kernel search (Angelelli et al. 2010) were applied to the problem and many valuable results were found.

## 3 Proposed method

In this section, a discrete model of *SEM* called *DEM* is proposed which is highly compatible with discrete space problems. To show the performance of the *DEM*, it is adapted for solving a well-studied discrete problem called *MKP* and the results are reported in next section.

### 3.1 Discrete electromagnetism-like mechanism

As mentioned in Sect. 2.1, the *SEM* utilizes some operators which have been inspired from the behavior of electrons in real word. The algorithm contains three main parts: calculating *charges* that are used for calculating *forces* and then particles are *moved* according to these forces. Here, a discrete model of *SEM* is proposed which works as *Algorithm* 2.

As understood from the algorithm, the *DEM* generally works the same as *SEM* but its operators work in the discrete spaces (see next sub-sections). Also, a new operator based on the principles of quantum mechanics (Stenholm and Suominen 2005) (called Annihilation/Creation ($A/A^{\dagger}$)) is proposed which helps the algorithm in sampling the search space (Solnon and Fenet 2006) and keeps the population from regenerating solutions. For more information on the effects of this operator, see Sect. 4.1.

**Algorithm 2** Solve a problem using *DEM*

*1) Initialize particles*
*2) While stopping criteria not emerged*
*a. Calculate objective values for all particles*
*b. Calculate charges*
*c. Calculate forces in the problem space*
*d. Move particles in the problem space according to forces*
*e. Apply Annihilation/Creation (A/A$^\dagger$)*
*3) Return the best particle as the best found solution*

## 3.2 Proposed operators

As presented in *Algorithm* 2, four main operators of the *DEM* are: *charge* calculation, *force* calculation, particles' *movement* and applying *A/A*$^\dagger$ on particles. It is clear that calculating the objective values is problem dependent. Consequently, the charge calculation procedure in *SEM* (which is only related to objective values, see Eq. 3) can be used in *DEM* and no discrete version for this operator is needed. In contrast, the other operators (Force calculation and Moving particles) should change to be consistent with discrete space problems. The discrete versions of these operators are presented in next sub-sections. Moreover, the *A/A*$^\dagger$ operator is described in the end of this sub-section.

### *3.2.1 Force calculation*

As put in Sect. 2.1, the effects (forces) from other individuals on a particle are calculated by subtraction (see Eq. 4). In fact, by subtracting the vectors (particles), their information is combined and the attained vector can be used as a good direction for movement. In a discrete problem space, sometimes it is not possible to add or subtract the individuals. Hence, the force calculation procedure in Eq. 4 is not appropriate for discrete space problems.

On the other hand, crossover operator generates new individuals with combining the information of parents and can work in discrete spaces. Therefore, to work in discrete spaces we can replace the additions and subtractions by crossover operator. Indeed, to compute the effect (force) of *i*th particle on *j*th particle in a discrete space, crossover operator is applied to these particles. *Algorithm* 3 presents the proposed force calculation procedure for the *DEM*. In fact, the algorithm calculates the total incoming forces on particle $i$ from all other particles ($j$). We have to note that the crossover is applied to particles according to a parameter called effect's ratio (*ER*) that is defined subsequently.

The effect's ratio ($ER_{ij}$) is calculated according to $q^i$ and $q^j$ (Eq. 3, charges of particles $i$ and $j$). In this paper, the following equation is proposed for calculating the $ER_{ij}$.

$$ER_{ij} = \frac{q^j}{q^i + q^j} \tag{9}$$

In this equation, $q^i$ is the charge of particle $i$. By using this equation, if particle $j$ is better than particle $i$ (in terms of objective value, note that the value of charges are

**Algorithm 3** Calculate forces on particle $y^i$

***Input:*** $y^i$, $\lambda$, $q^i$

***Output:*** $F^i$: *total forces on* $y^i$

*1) Consider* $F^i$ *as a vector with the length equal to the length of* $y^i$ *and its values are set to 0 for initialization*

*2) For each particle j (* $j \neq i$*)*

*a. If rand()* $< \lambda$

*i. Calculate* $ER_{ij}$ (Eq. 9)

*ii.* $F^i = (F^i \oplus y^j)_{ER_{ij}}$

*3) Return the value of* $F^i$

calculated according to objective values), the effect of this particle on particle $i$ is increased and vice versa. Also, in the *Algorithm* 3, $(F^i \oplus y^j)_{ER_{ij}}$ shows crossover operator which combines vector $F^i$ and vector (particle) $y^j$ according to the effect's ratio of particle $i$ on particle $j$ and the *rand()* function generates random values in the interval [0,1]. Moreover, the force from particle $i$ on particle $j$ is considered in calculation with probability of $\lambda$. The value of this parameter has a very essential effect on the convergence speed and execution time of the algorithm.

It is worth mentioning that the model of the crossover is problem dependent but it should be designed in such a way that it combines particles according to the *ER* (Eq. 9). As an example, in Sect. 3.3.3 a specific model of uniform crossover called Effects-based Uniform Crossover (EUX) for the *MKP* is proposed which exerts the effect's ratio to combine particles.

### *3.2.2 Move*

Another important part of the *DEM* is particle's movement. In *Algorithm* 2, it was shown that the forces calculated by addition (Eq. 5) are used for particles' movement. In the *DEM*, the adding operator is replaced by crossover operator which is compatible with discrete spaces. In fact, the crossover operator is used to move particles according to their forces. The moving operator is applied to particles with probability $P_{\text{move}}$.

The algorithm shows how calculated forces are used to move the particles. The force $F^i$ and particle $y^i$ are combined together using crossover operator. In our implementations for solving the *MKP*, the standard uniform crossover was used. This leads to a balanced combination between a particle and its incoming resultant forces. Also, the procedure is done with probability $P_{\text{move}}$.

### *3.2.3 Annihilation/creation (A/A*$^{\dagger}$*)*

Assume that *DiffNum* is the number of unique solutions and *TotalNum* is the number of all generated solutions from first generation so far, then the re-sampling ratio will be defined as *(TotalNum − DiffNum)/TotalNum*. If the value for this formula is close to 0, then we can say that the algorithm is efficiently searching the problem space, i.e., a few duplicate solutions are generated, while value close to 1 indicates that the search is in stagnant condition, i.e., a few new solutions are generated. To

**Algorithm 4** Move particle $y^i$

***Input:*** $\boldsymbol{F^i}$, $\boldsymbol{P_{move}}$
***Output:*** $\boldsymbol{y^i}$
*1) If rand() <* $P_{move}$
*a.* $y^{i*} = (y^i \oplus F^i)$
*b. If objective value for* $y^{i*}$ *is bigger than the objective value for* $y^i$ *then replace it by* $y^{i*}$
*2) Return* $y^i$

**Algorithm 5** $A/A^\dagger$

***Input: x (number of successive iterations that the particle has not been improved),*** $\boldsymbol{y^i}$
***Output:*** $\boldsymbol{y^i}$
*1) If rand() <* $P_{A/A^\dagger}(X = x)$
*a. Reinitialize* $y^i$
*2) Return* $y^i$

decrease the value of the re-sampling ratio, the $A/A^\dagger$ operator is proposed. Based on the principles of quantum mechanics, a quantum particle can be annihilated but it is created in another place in its world (Stenholm and Suominen 2005). Here, this approach is used and each particle in the population is annihilated with probability $P_{A/A^\dagger}(X)$ and is created in a randomly determined new place in the problem space ($X$ is a random variable). In this paper, $X$ is considered as the number of successive generations that the particle has not been improved in. Also, a probability density function (*PDF*) is needed to determine the value of $P_{A/A^\dagger}(X)$. There are two important points that should be noted for determining the *PDF*. First, the value of $P_{A/A^\dagger}(X)$ should be very small for small $X$s to retain the lately created particles from annihilation. Also, it should be increased (from near 0 to close to 1) by $X$, i.e., a particle which has not been improved for last $x$ generations has a smaller probability to be annihilated than a particle which has not been improved for last $x + 1$ generations. As an example, for solving the *MKP*, a *PDF* is proposed which is described in Sect. 3.3.4. This operator can efficiently decrease re-sampling ratio and hence help the algorithm to seek the search space thoroughly. Also, it can strongly decrease the similarity ratio (see Sect. 3.3.4)

In this algorithm, *Reinitialize* operator locates the particle to a new random position in the feasible space. Now, all operators have been introduced for the *DEM*. The following algorithm shows the *DEM* for solving discrete problems (Algorithm 6).

## 3.3 Proposed *DEM* for the *MKP*

In this section, a *DEM*-based approach is proposed for solving the *MKP*. First, we describe the used coding scheme for presenting solutions of the *MKP*. Afterwards, the calculation of objective value and the used heuristic is explained and then

**Algorithm 6** *DEM*

***Input: problem***
***Output: solution***
*4) Initialize particles*
*5) While stopping criteria not emerged*
*a. Calculate objective values for all particles*
*b. Calculate charges (Eq. 3)*
*c. Apply Algorithm 3 on all particles for Calculating forces*
*d. Apply Algorithm 4 on all particles to Move them*
*e. Apply Algorithm 5 on all particles to determine whether they should be annihilated*
*6) Return the best particle as the best found solution*

operators for movement, force calculation (crossover) and mutation are introduced. Moreover, the $A/A^{\dagger}$ operator and the used local search are elaborated on.

### *3.3.1 Coding scheme*

The bit string is one of the most used coding schemes for solving the *MKP* (Chu and Beasley 1998). In this paper, this presentation is used for the particles. In fact, each particle consists of a string of 0/1 s and each element of each particle indicates whether the corresponding item should be picked. Figure 2 shows a particle for a problem with 10 items.

As it seems from the figure, the example particle picks the first, third, fourth and last items and the other items are not selected.

### *3.3.2 Objective value*

The objective value for each particle is calculated according to the summation of prices of the picked items. Because the picked items may violate the constraint condition, a validation phase is needed. In this phase, heuristic information on the basis of the pseudo-utility ratios is used and defined by:

$$d_i = \frac{p_i}{\sum_j s_j w_{ij}} \tag{10}$$

In this equation, $s_j$ is the shadow price of the $j$th constraint in the linear programming relaxation of the original *MKP* and $d_i$ is the density of $i$th item. We obtain the weights $s_j$ by using a method by Pirkul (1987), in which the LP relaxation of the original *MKP* is solved and the values of the dual variables are viewed as the


**Fig. 2** An example of coding scheme for presenting a solution. In this paper, this presentation was used for particles

weights. The validation operator is inspired from the idea of Chu and Beasley (1998) which consists of two phases. The first phase (called DROP) examines each bit of the solution string in increasing order of $d_i$ and changes a bit from one to zero if feasibility is violated. The second phase (called Adding phase) reverses the process to choose an item according to its density.

### *3.3.3 Crossover and mutation*

In this paper, a new version of uniform crossover for the *MKP* (called Effects-based Uniform Crossover (*EUX*)) is proposed which works according to the effect's ratio of particles. In uniform crossover, two parents have a single child. Each bit in the child solution is created by copying the corresponding bit from one or the other parent, chosen according to a binary random number generator [0, 1]. If the random number is a 0, the bit is copied from the first parent, if it is a 1, the bit is copied from the second parent. By contrast, in the proposed *EUX* the random number generator generates a number in the interval [0, 1]. If this number is smaller than $ER_{ab}$ (Eq. 9), the bit is chosen from the first parent; else it is selected from the other parent. The pseudo-code of this crossover has been shown in *Algorithm 7*.

In this algorithm, $a$ and $b$ are two individuals and $a_i$ and $b_i$ are their elements. The value of $ER_{ab}$ is calculated by the notes that were made in Sect. 3.2.1 (Eq. 9). Also, a max-min approach is proposed which decreases the re-sampling and similarity ratio. In this approach, the value of $ER_{ab}$ is bounded in the interval *[p1, p2]* to give chance to worst particles for participating in crossover procedure. This approach highly increases the exploration of new possible better solutions. We call this crossover as *MMEUX* (Max Min Effects-based Uniform Crossover). The effect of this approach on the re-sampling and similarity ratio is analyzed in next sections.

After the crossover phase, the Mutation is performed to mutate some randomly selected bits with probability of $P_{\text{mutation}}$. The mutation is applied to population to refrain it from getting stock on local optima. The used mutation for the *MKP* was as follow (Chu and Beasley 1998):

In each generation, each element of each particle is reversed with a specified probability ($P_{\text{mutation}}$) and the resulted particle is validated and fed to the population. In this paper, the value of $P_{\text{mutation}}$ was related to population size (PS) as $P_{\text{mutation}} = C_{\text{mutation}}/PS$ where $PS$ is the population size (number of particles) and $C_{\text{Mutation}}$ is a coefficient which controls the value of $P_{\text{mutation}}$. This relation facilitates the parameter setting presented in Sect. 4.1.

**Algorithm 7** EUX

***Input: a, b, $ER_{ab}$***

*1)* For all elements in $a$ and $b$ ($a_i$ and $b_i$)

*a.* If *rand()* $< ER_{ab}$

*i.* $C_i = a_i$

*b.* Else

*i.* $C_i = b_i$

*2)* Return $C$ as a new individual

### 3.3.4 Annihilation/creation ($A/A^{\dagger}$)

As it was mentioned in Sect. 3.2.3, an annihilation/creation operator can be very efficient in sampling the problem space. Also, a *PDF* is needed to determine the probability of applying the operator to a particle. Also, a random variable ($X$) of the *PDF* was defined in that section which was the number of successive generations that the particle has not been changed. Here, a *PDF* is proposed for this procedure which is shown in Eq. 11.

$$P_{A/A^{\dagger}}(X = x) = \frac{1}{1 + e^{-a(x-c)}} \tag{11}$$

As it is clear from the equation, the value of the probability is increased by $X$, which is desirable according to Sect. 3.2.3. To design an appropriate function, the variables $a$ and $c$ should be adjusted. Here, the probability of small $X$s should be very small to keep the newly generated particles from annihilation. Hence, we considered that the value of the probability for $x = l_1$ is $p_1$, i.e., $P_{A/A^{\dagger}}(X = l_1) = p_1$. This means that the probability of applying the $A/A^{\dagger}$ operator on a particle which has not been improved for last $l_1$ generations is $p_1$. Because the function is increased by $X$, the value of the probability is less than $p_1$ when $x < l_1$. Also, the value for the probability should be big when $x$ is big enough. Therefore, for $x = l_2$ the value of the probability is considered as $p_2$, i.e., $P_{A/A^{\dagger}}(X = l_2) = p_2$. By using this information and substituting two pairs $(l_1, p_1)$ and $(l_2, p_2)$ in Eq. 11, we can say that:

$$a = \frac{\ln\left(\frac{p_2(1-p_1)}{p_1(1-p_2)}\right)}{l_2 - l_1} \tag{12}$$

and

$$c = \frac{\ln\left(\frac{1}{p_1} - 1\right)}{a} + l_1 \tag{13}$$

In these equations *ln* is the natural logarithm. In this paper, we considered the value of $p_1$ as 0.01, $p_2$ as 0.5, $l_1$ as $l$ and $l_2$ as $2l$ which means the probability of annihilating a particle that has not been improved for last $l$ generations is 0.01 while this probability for a particle which has not been improved for last $2l$ generations is *0.5*. In this case, the value of $a$ and $c$ are calculated via following equations:

$$a = \frac{\ln\left(\frac{0.5(1-0.01)}{0.01(1-0.5)}\right)}{2l - l} = \frac{\ln(99)}{l} \tag{14}$$

and

$$c = \frac{\ln(99)}{a} + l = 2l \tag{15}$$

The value of $l$ is adjusted according to the number of particles ($l = PS \times C_l$) in the population and is examined in Sect. 4.1.

### 3.3.5 Local search

In this paper, we used a simple random local search method which fillips $0.04n$ variables each time where $n$ is the number of objects. In fact, in a 100 objects problem, 4 randomly selected variables from a solution are filliped from 1 to 0 or 0 to 1, the new solution is validated if necessary. If the new generated solution is better, the original solution is replaced with the new one. The local search method is performed 1,000 times for each solution (Kong et al. 2008) and it is applied with probability 0.01 on each particle. Our experiments showed that the bigger values for the probability and number of fillips have no significant effect on the performance of the algorithm.

## 4 Experimental results

In this section, the experimental results are reported. First, the used test benches are introduced. Next, the parameters of the *DEM* are set and then the *DEM* is applied to standard test benches. Finally, the results are reported and compared with several new methods. For all tests, the standard test benches which have been stemmed from the OR-Library by Beasley (1990) are used. The mentioned library contains 9 test benches; each of them includes 30 instances. Table 1 shows the information of these problems.

Each *m–n* problem set (test bench) contains 30 problems which are split into three groups in terms of the tightness ratio. The first 10 problems for each test bench (problem 00 to 09 inclusive) include the problems with tightness ratio 0.25, the second 10 problems (problem 10 to 19 inclusive) consist of the problems with tightness ratio 0.5 and the last 10 problems (problem 20 to 29 inclusive) contains the tightness ratio 0.75. Hereafter, we address the instances by the notation *m–n:i* which refers to the *i*th instance of the test bench with *m* constraints and *n* items. As an example, the notation 5–100:08 points to the 9th instance of the test bench with 5 constraints and 100 items. In addition, the notation *m–n(a)* refers to 10 problems of the test bench *m–n* with tightness ratio *a*. For example, the notation 30–250(0.5) addresses the second 10 problems in the test bench 30–250.

**Table 1** Nine used test benches for OR-Library

| Problem set | Number of constraints-items (*m–n*) | Number of instances | Problem set | Number of constraints-items (*m–n*) | Number of instances |
|---|---|---|---|---|---|
| 1 | 5–100 | 30 | 2 | 5–250 | 30 |
| 3 | 5–500 | 30 | 4 | 10–100 | 30 |
| 5 | 10–250 | 30 | 6 | 10–500 | 30 |
| 7 | 30–100 | 30 | 8 | 30–250 | 30 |
| 9 | 30–500 | 30 | – | – | – |

Each test bench is contained 30 problems

### 4.1 Parameter setting

There are several parameters for the *DEM* that should be set, that are, $P_{\text{move}}$, $P_{\text{mutation}}$, $\lambda$, $l$ (for $A/A^{\dagger}$ operator) and *PS* (population size). Also, the effects of *MMEUX* (against *EUX*) and $A/A^{\dagger}$ operator should be tested.

To find the best values for $P_{\text{move}}$, $P_{\text{mutation}}$, $\lambda$ and $l$ (for $A/A^{\dagger}$ operator), these values were changed and the *DEM* was applied to 6 problems (5–100:00, 05, 10, 15, 20 and 25) from OR-Library, 25 times, and the average of the objective values were reported. Table 2 shows the results.

Each row of the table shows the setting procedure for the parameter that has been shown in the first column. The second column presents the relation that has been considered for each parameter. As an example, the value of $l$ is linearly related to *PS* by a coefficient called $C_l$ ($l = PS \times C_l$). In the third column (Interval: Step size), the intervals used for altering the parameters have been depicted. Also, the step sizes for changing the value of the parameters have been presented. The fourth column shows the values of other parameters in each setting procedure. The best found value for the parameters has been indicated in the fifth column labeled as "Best value". In fact, these values are the best effective values which could maximize the performance of the *DEM* in that test. Also, the last column shows the best found average of solutions for 6 mentioned problems. Note that the *DEM* was applied to each problem 25 times and the average of found solutions for all runs on all 6 problems has been shown in the table.

The curves in Fig. 3 show the results of changing the parameters in the mentioned intervals. In each curve, the value of each parameter has been shown in $x$ axis while the corresponding average of objective values (over 25 runs on 6 mentioned problems) has been presented in $y$ axis.

Figure 3 shows the variations of values $P_{\text{move}}$, $C_{\text{mutation}}$, $\lambda$ and $C_l$ in the mentioned intervals in Table 2. Also, the $y$ axis of each curve shows the average of objective values. Figure 3a shows that the best performance of the *DEM* appears

**Table 2** Results of parameter setting

| | Relation | Interval: step size | Other parameters | | | | | Best value | Average of objective values |
|---|---|---|---|---|---|---|---|---|---|
| | | | $P_{\text{move}}$ | $P_{\text{mutation}}$ | $\lambda$ | $l$ ($A/A^{\dagger}$) | PS | | |
| $P_{\text{move}}$ | – | $P_{\text{move}} = [0.1, 0.95]:0.05$ | – | 0 | 1 | Infinite* | 50 | 0.9 | 42176.58 |
| $P_{\text{mutation}}$ | $C_{\text{Mutation}}/PS$ | $C_{\text{Mutation}} = [0.5, 11.5]:0.5$ | 0.9 | – | 1 | Infinite | 50 | 8.5/*PS* | 42206.26 |
| $\lambda$ | – | $\lambda = [0.1, 1]: 0.1$ | 0.9 | 8.5/PS | – | Infinite | 50 | 0.6 | 42219.32 |
| $l$ ($A/A^{\dagger}$) | $PS \times C_l$ | $C_l = [0.01, 0.25]:0.01$ | 0.9 | 8.5/PS | 0.6 | – | 50 | $0.06 \times PS$ | 42222.33 |

Each row shows setting procedure for parameters, i.e., $P_{\text{move}}$, $P_{\text{mutation}}$, $\lambda$ and $l$

* The value "Infinite" for $l$ means that the particles are not annihilated

when the value of $P_{\text{move}}$ is 0.9. Figure 3b indicates that the value of $C_{\text{mutation}}$ should be 8.5 ($P_{\text{mutation}} = 8.5/PS$) and Fig. 3c implies that the best value for $\lambda$ is 0.6. From Fig. 3d it is concluded that the best value for $C_l$ is 0.06.

Hence, by using the Eqs. 14 and 15, we can say that:

$$a = \frac{\ln(99)}{l} \overset{l=0.06PS}{\longrightarrow} a = \frac{\ln(99)}{0.06PS} \simeq \frac{76.58}{PS} \tag{16}$$

and

$$c = 2l = 0.12 \times PS \tag{17}$$

In these equations $PS$ is the population size. For example, for a population with 50 particles the best performance appears when the value of $l$ is $3(l = 0.06 \times 50)$. Hence, according to Eqs. 11, 16, and 17, the $PDF$ for random variable $X$ is as follow:

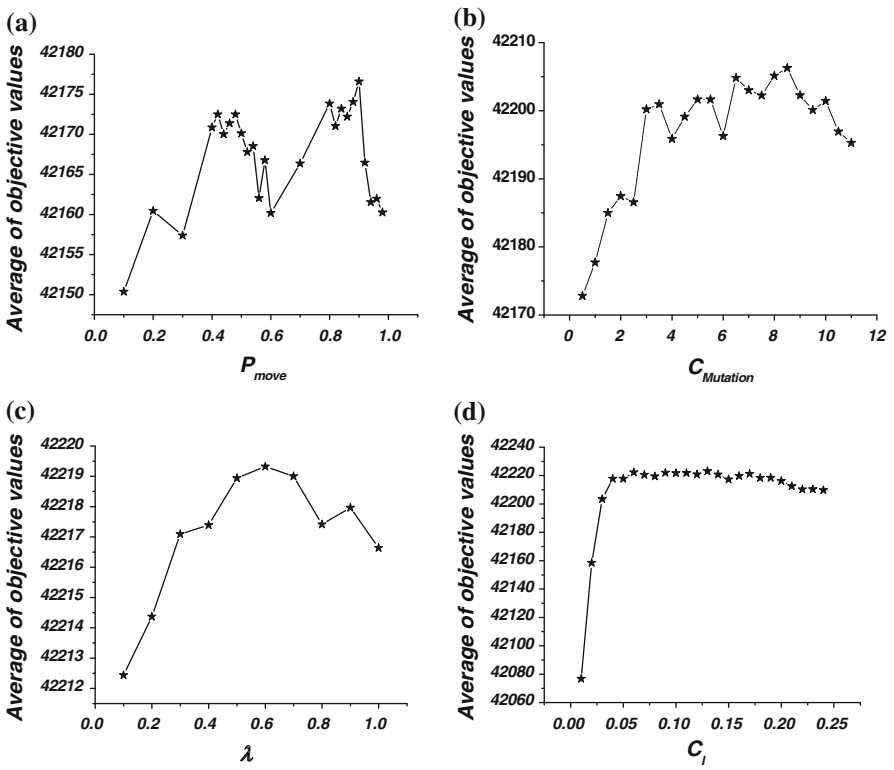


**Fig. 3** Each curve shows the value of parameter in $x$ axis while the average of objective values for 6 standard problems over 25 runs has been shown in $y$ axis (**a**) variation of $P_{\text{move}}$ in the interval [0.1, 0.95], (**b**) variation of $C_{\text{mutation}}$ ($P_{\text{mutation}} = C_{\text{mutation}}/PS$) in the interval [0.5, 11.5] with the step size 0.5, (**c**) variation of $\lambda$ in the interval [0.1, 1], and (**d**) variation of $C_l$ ($l = PS \times C_l$) in the interval [0.01, 0.25]. The average of objective values were highly decreased for $C_l > 0.25$ and has not been included in the figure

$$P_{A/A^{\dagger}}(X = x) = \frac{1}{1 + e^{-1.53(x-6)}} \tag{18}$$

By using this equation, a particle which has not been improved for last 6 generations is annihilated and created again with probability of about 0.5. Note that this equation is used when the population size is 50.

Finally, the value of $PS$ (population size) is examined. For this purpose we considered that the value of $PS$ is linearly related to number of items. Thus, we can say that $PS = n \times C_{PS}$ where $n$ is the number of items and $C_{PS}$ is a constant. The value of $C_{PS}$ is changed in the interval [0.1, 1.2] with the step size 0.1 for test benches 5–250, 5–500, 10–250, 10–500, 30–250, and 30–500. These problems were selected for this test because their number of items and number of constraints were different. The results are shown in Fig. 4. In this test, the termination condition was the maximum number of evaluations which was fixed to 10,000.

From the figure, it is obvious that the algorithm has the best performance when the value of $C_{PS}$ is 1, i.e., the $PS$ is equal to number of items. In addition, the number of constraints has not affected the $C_{PS}$. Also, the bigger values for $C_{PS}$ have not affected the performance of the algorithm. Hence, in all further tests, the $PS$ is considered as $n$ (number of items).

To summarize the setting of parameters, the following values were attained and are used hereafter.

- PS *(Population size)*: *n (number of items)*
- $P_{move}$:0.9
- $P_{mutation}$: 8.5/*n*
- $\lambda$:0.6
- $P_{A/A^{\dagger}}(X = x) = \frac{1}{1+e^{-a(x-c)}} \overset{l=0.06n}{\longrightarrow} P_{A/A^{\dagger}}(X = x) = \frac{1}{1+e^{-\frac{76.58}{n}(x-0.12n)}}$

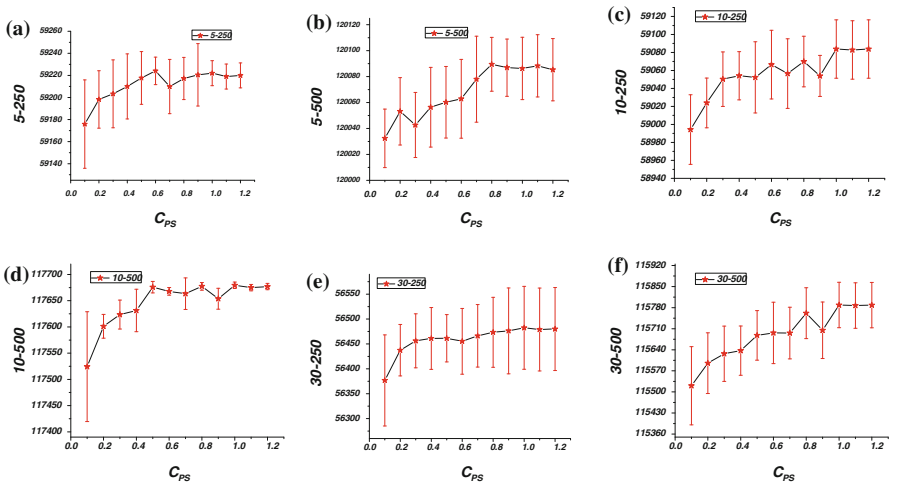


**Fig. 4** The results of applying the *AMMDEM* on 6 problems (**a**) 5–250:00, (**b**) 5–500:00, (**c**) 10–250:00, (**d**) 10–500:00, (**e**) 30–250:00, and (**f**) 30–500:00. Each point in each graph is the average over 10 runs. Also, the STDs have been presented. The $x$ axis is the $C_{PS}$ ($PS = n \times C_{PS}$) while $y$ axis is the objective value

Also, to show the effects of $A/A^{\dagger}$ and *MMEUX* on re-sampling, the *DEM* (*DEM* with *EUX* crossover), *MMDEM* (*DEM* with *MMEUX* crossover) and *AMMDEM* (*DEM* with *MMEUX* crossover and $A/A^{\dagger}$ operator) were applied to the problem 5–100:00 and the re-sampling and similarity ratios have been reported in Fig. 5.

As Fig. 5a shows, it is obvious that, although the $A/A^{\dagger}$ operator could strongly reduce the re-sampling ratio, the max-min approach did not highly affect this ratio. We should note that the parameters for this test was as mentioned earlier. Furthermore, a similarity ratio is defined which shows the diversification in each generation. In fact, if *DiffNum* is considered as the number of different particles in the current run and *PS* be the population size, then *(PS-DiffNum)/PS* is called the similarity ratio. Note that the similarity ratio is calculated in each generation while the re-sampling ratio is calculated in all previous steps. Figure 5b shows the effects of *MMEUX* and $A/A^{\dagger}$ on this ratio where problem 5–100:00 was considered as an example. The results (each point in the graph) are the averages over 25 runs. It is clear that the similarity ratio for the *AMMDEM* is smaller than *MMDEM* and *DEM*. Also, the *MMEUX* crossover has decreased the similarity ratio.

## 4.2 Simulation results

In this sub-section, the proposed method is compared with four methods called *BAS* + *ls* (Kong et al. 2008), *FSC* (Hanafi and Wilbaut 2008), *DMMAS* (Ke et al. 2010), *Fixed-bucket*-I(0.1) (Angelelli et al. 2010), and a well-known traditional algorithm based on *GA* (Chu and Beasley 1998). The implementation was done under the Visual Studio.Net 2008 environment with the *C#* language and all tests were performed on a personal computer with 1.66 GHZ of *CPU* and 2 GB of *RAM*. To show the performance of the proposed approach, first, the results of the *DEM*,

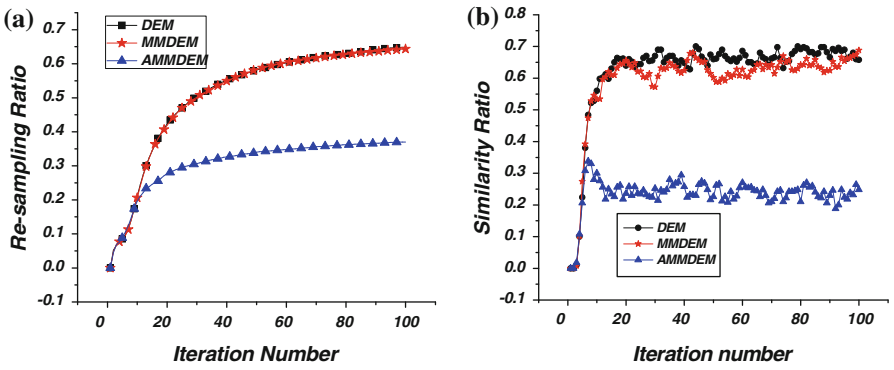


**Fig. 5** The test was performed on the test problem 5–100:00 (**a**) the re-sampling ratio is decreased when the $A/A^{\dagger}$ operator is adjoined to the *DEM*, (**b**) the similarity ratio is decreased when the $A/A^{\dagger}$ operator is added to the *DEM*. Also, the *MMEUX* could affect the *DEM* in terms of the similarity ratio

*AMMDEM* and *AMMDEM* + *ls* (the *AMMDEM* that has been combined with the local search that was introduced in Sect. 3.3.5) are examined (First test). Afterwards, the *AMMDEM* and the *DMMAS* are juxtaposed. In this test, the algorithms (*AMMDEM* and *DMMAS*) are applied to 10 first instances of 5–100 and 10–100 test benches (5–100(0.25) and 10–100(0.25)). Also, both of them are applied to all problems in the test bench 5–500 (Second test). Then, the *AMMDEM* + *ls* is compared with the *BAS* + *ls* (Kong et al. 2008) in terms of the average of found solutions and the *CPU* time for finding the best values (Third test). In this case, the algorithms (*BAS* + *ls* and *AMMDEM* + *ls*) are applied to 5–100 (30 problems) and 10–100 (30 problems). Finally, a comparison among *AMMDEM* + *ls*, *Fixed-bucket*-I(0.1) (Angelelli et al. 2010), *FSC* (Hanafi and Wilbaut 2008), *CPLEX* and the $GA_{CB}$ (Chu and Beasley 1998) (a GA based method that was proposed by Chu and Beasley) is done (Fourth test). It is worth mentioning that the *Fixed-bucket-I(a)* is a heuristic method based on kernel search. In Angelelli et al. (2010), three types of this search method was introduced called *Fixed-bucket*-I (1,0.2,0.1). The *Fixed-bucket*-I(0.1) is used for comparison because this type converges much faster to slightly weaker gap percentages than other two types (*Fixed-bucket*-I(0.2,1)). In the latter comparison, the results of all 270 problems are reported and these results are compared in terms of the percentage of gap from the optimum values of the *LP-relaxation* (since the optimum solution values are not known for all the instances) and *CPU* time. These methods are used for comparison purposes since they are all population-based, have obtained valuable results for the *MKP* and are recently proposed (except for $GA_{CB}$). We should keep this point in mind that the parameters of all runs were as mentioned in the previous section. Also, the best found objective values for some problems in OR-Library are listed in "Appendix".

### *4.2.1 First test*

Table 3 demonstrates the results of the *AMMDEM*, *MMDEM*, and *DEM* when they were applied to test benches 5–100, 10–100, and 5–250.

In each test, the mentioned algorithms were applied to all 30 problems from the test benches 5–100, 10–100 and 5–250 (90 instances in sum). The depicted results are the average over 20 runs for each problem (1,800 runs in sum). The algorithms were terminated after 10,000 evolutions, i.e., the maximum generation for the 5–100, 10–100 and 5–250 test benches was 100, 100 and 40, respectively. Note that the population size for each test was (as it was mentioned in the previous section) set to the number of items ($n$). From the table it is seen that the *AMMDEM* excels the *DEM* in terms of the average of found solutions. Also, the local search significantly improved the performance of the *AMMDEM*. It is worth noting that the *AMMDEM* and *AMMDEM* + *ls* could find the best known solutions for all listed test benches in Table 3 except for the case 5–250:29. In this case, the best reported price was 154,662 by Chu and Beasley (1998) while the *AMMDEM* + *ls* found a better solution with the price 154,668.

**Table 3** The average of objective values when the *DEM*, *MMDEM*, and *AMMDEM* are applied to all problems in test benches 5–100, 5–250, and 10–100

| Used method | Test bench | | |
|---|---|---|---|
| | 5–100 | 10–100 | 5–250 |
| *DEM* | 42601.41 | 41533.98 | 107032 |
| *AMMDEM* | 42610.03 | 41541.33 | 107040.9 |
| ***AMMDEM + ls*** | **42639.72** | **41603.27** | **107601** |

Bold values shows the best obtained results by *DEM*, *AMMDEM*, and *AMMDEM + ls*

#### *4.2.2 Second test*

In the second test, the *AMMDEM* is compared with a new method called *DMMAS*. In this test, 10 first instances of test bench 5–100(5–100: (0.25)), 10 first instances of test bench 10–100 (10–100(0.25)) and 5 first instances of test bench 5–500 are considered. The results of the algorithms are reported in Table 4.

The averages have been calculated over 25 runs for the *AMMDEM* and 50 runs for the *DMMAS*. Also, the stopping criterion for both methods was the predefined maximum evaluations (10,000). As it seems from the table, the *AMMDEM* surpassed the *DMMAS* in 15 cases over all 25 reported cases in terms of the average of found solutions. In addition, in 14 cases, the STD (standard deviation) of the *AMMDEM* was better than the STD of *DMMAS* indicating the stability of the proposed algorithm. Furthermore, both algorithms found the best known optimum values for the problem set 5–100(0.25) and 10–100(0.25). For the test bench 5–500, the *AMMDEM* could find better solutions in 3 cases (5–500:00, 02, and 03). Also, the results of applying both methods (*AMMDEM* and *DMMAS*) on all problems of the test bench 5–500 have been shown in Table 5.

The *AMMDEM* was applied 25 times on each problem from the 5–500 test benches and the stopping criterion for both methods was the same (10,000 evaluation). As the table shows, the *AMMDEM* attained better average of solutions. Also, it excelled the *DMMAS* in terms of the average of best found solutions for these problems. It is worthwhile to note that the *AMMDEM* took 44 s in average to find the best solution in each run. To evaluate the efficiency of the *AMMDEM + ls*, the method is compared with three methods which all of them use a local search (*BAS + ls*, $GA_{CB}$ and *FSC*).

#### *4.2.3 Third test*

In this test, the *AMMDEM + ls* is compared with *BAS + ls*. The local search for the *BAS + ls* was the same as the local search for *AMMDEM + ls* (see Sect. 3.3.5). Table 6 shows the results of these methods when they were applied to the 5–100 and 10–100 test benches. The $CPU_{\text{avr}}$ shows the average of time that an algorithm takes to reach the final best solution for the first time.

The *AMMDEM + ls* was tackled each problem 25 times and the average of best found solutions, average of solutions, and $CPU_{\text{avr}}$ times (in second) have been reported in the table. The *AMMDEM + ls* found the best known solutions in all 60 cases while

**Table 4** The results of comparison between the proposed method and *DMMAS*

| Problems | Best known solutions (Boussier et al. 2010) | AMMDEM | | DMMAS | |
|---|---|---|---|---|---|
| | | Best found solutions | Average of found solutions | Best found solutions | Average of found solutions |
| 5:100-00 | 24,381 | **24,381** | **24,373 (17.2)** | **24,381** | 24,362 (23.8) |
| 5:100-01 | 24,274 | **24,274** | 24,266.4 (13.6) | **24,274** | **24,273 (6.2)** |
| 5:100-02 | 23,551 | **23,551** | 23,526 **(6)** | **23,551** | **23,540** (7.2) |
| 5:100-03 | 23,534 | **23,534** | **23,483.1** (22.4) | **23,534** | 23,482 **(14.9)** |
| 5:100-04 | 23,991 | **23,991** | **23,960.6 (10.8)** | **23,991** | 23,954 (10.8) |
| 5:100-05 | 24,613 | **24,613** | 24,598.8 (11.6) | **24,613** | **24,608 (6.3)** |
| 5:100-06 | 25,591 | **25,591** | **25,591** (21) | **25,591** | **25,591 (0)** |
| 5:100-07 | 23,410 | **23,410** | 23,361.5 (14.8) | **23,410** | **23,404 (13.3)** |
| 5:100-08 | 24,216 | **24,216** | **24,212 (6)** | **24,216** | 24,211 (5.9) |
| 5:100-09 | 24,411 | **24,411** | 24,400.8 (20.4) | **24,411** | **24,406 (13.8)** |
| 10:100-00 | 23,064 | **23,064** | **23,050.7 (2.1)** | **23,064** | 23,045 (19.6) |
| 10:100-01 | 22,801 | **22,801** | 22,741.2 **(17.6)** | **22,801** | **23,742** (40.8) |
| 10:100-02 | 22,131 | **22,131** | **22,104.4** (32.5) | **22,131** | 22,091 **(29.7)** |
| 10:100-03 | 22,772 | **22,772** | **22,725 (28.6)** | **22,772** | 22,710 (37.6) |
| 10:100-04 | 22,751 | **22,751** | **22,618.8 (31.2)** | **22,751** | 22,617 (43.9) |
| 10:100-05 | 22,777 | **22,777** | **22,667.6 (31.6)** | **22,777** | 22,663 (40.3) |
| 10:100-06 | 21,875 | **21,875** | 21,794 (10) | **21,875** | **21,826** (28.4) |
| 10:100-07 | 22,635 | **22,635** | 22,539.5 (7.5) | **22,635** | **22,557** (31) |
| 10:100-08 | 22,511 | **22,511** | 22,394 (18) | **22,511** | **22,409 (17.3)** |
| 10:100-09 | 22,702 | **22,702** | **22,702 (0)** | **22,702** | 22,696 (32.7) |
| 5:500-00 | 120,148 | **120,118** | **120,087.1 (26.7)** | 120,116 | 120,043 (31.2) |
| 5:500-01 | 117,879 | 117,836 | **117,814.2 (18.3)** | **117,854** | 117,777 (33.3) |
| 5:500-02 | 121,131 | **121,109** | **121,089.5 (16.5)** | 121,102 | 121,023 (35.4) |
| 5:500-03 | 120,804 | **120,785** | **120,742.9 (17.4)** | 120,778 | 120,707 (32.6) |
| 5:500-04 | 122,319 | **122,319** | **122,282.2** (34.5) | **122,319** | 122,249 **(32.1)** |

The termination criterion for both methods was the maximum evaluations (10,000). Bold underlined results are the best results in that column. Bold results are the results in which both algorithms found the same result

the *BAS* + *ls* could not find the best known optimum for the problem 10–100:26. Also, the average of execution time for the test bench 5–100 was 13.4(s) for the *AMMDEM* + *ls* while this time was 50.9(s) for the *BAS* + *ls* (77.5% improvement). Besides, the *AMMDEM* + *ls* found the reported solutions for the test bench 10–100 in 32.7 (s) in average while the *BAS* + *ls* found the solutions in 86.8 (s) (62% improvement).

### *4.2.4 Forth test*

Finally, the *AMMDEM* + *ls* is compared with the *Fixed-bucket*-I(0.1), *FSC, CPLEX* of Ilog (Version 9) and $GA_{CB}$. Table 7 shows the results of these algorithms when they were applied to all 270 cases from OR-Library.

**Table 5** The *AMMDEM* and the *DMMAS* have been compared in terms of the average and best found objective values when they were applied to all problems in test bench 5–500

| Tightness ratio | AMMDEM | | DMMAS | |
|---|---|---|---|---|
| | Average of found objective value | Best found objective value | Average of found objective value | Best found objective value |
| 0.25 | ***120,574.4*** | ***120,605.6*** | 120,541.1 | 120,605.3 |
| 0.5 | ***219,463.5*** | ***219,496.8*** | 219,438.8 | 219,496.6 |
| 0.75 | ***302,318.5*** | ***302,354.3*** | 302,303.3 | 302,353.3 |
| Average | ***214,118.8*** | ***214,152.23*** | 214,094.4 | 214,151.73 |

The reported results are the averages over 25 runs for *AMMDEM* and 50 runs for *DMMAS*. Bold underlined italicized results show the best found results by *DMMAS* and *AMMDEM* + *ls*

**Table 6** The comparison results of the *AMMDEM* + *ls* and *BAS* + *ls* when they are applied to test benches 5–100 and 10–100

| | 5–100 | | | 10–100 | | |
|---|---|---|---|---|---|---|
| | Average of found objective value (Aver.) | Best found objective value (Aver.) | $CPU_{avr}$ | Average of found objective value (Aver.) | Best found objective value (Aver.) | $CPU_{avr}$ |
| *AMMDEM* + *ls* | ***42,639.72*** | ***42,640.3*** | ***13.47*** | ***41,603.27*** | ***41,606.03*** | ***32.71*** |
| *BAS* + *ls* | 42,639.51 | ***42,640.3*** | 50.94 | 41,601.11 | 41,605.23 | 86.88 |

The results are average over all problems. Each algorithm was applied to the problems 25 times. Bold underlined italicized results show the best found results by *AMMDEM* + *ls* and *BAS* + *ls*. Bold italicized results are the results in which both algorithms found the same result

The termination criterion for the *AMMDEM* + *ls* was the maximum evaluation (10,000) in this test. It is obvious from the table that the proposed method excelled all other methods in terms of the $CPU_{avr}$ which was defined previously. Also, Fig. 6a shows the average of gaps for the listed algorithms in Table 7 and Fig. 6b exhibits the average of $CPU_{avr}$ for all *MKPs* in OR-Library.

From the figure, it is clear that the *AMMDEM* + *ls* surpasses *CPLEX, FSC*, and $GA_{CB}$ in terms of the $CPU_{avr}$ (79, 79, and 83% improvement in comparison with the *CPLEX*, *FSC*, and $GA_{CB}$, respectively) and percentage of gap (0.2, 1, and 1.4% improvement in comparison with the *CPLEX*, *FSC*, and $GA_{CB}$, respectively). We have to mention that the best solutions for 5–500, 10–500, and 30–500 are achieved by the *Fix* + *LP* + *TS* algorithm of Vasquez and Vimont (2005). However, their computation took inordinately long times, with some instances solved for more than 3 days (about 16 h in average for solving each problem).

In addition, the proposed algorithm works much faster to competitive results (in terms of Gap percentage) when it is compared to *Fixed-bucket*-I(0.1).

## 5 Conclusion

The Standard Electromagnetism-Like Mechanism (*SEM*) is one of the population-based optimization methods which was proposed and successfully applied to

**Table 7** The comparison results among *AMMDEM + ls*, *Fixed-bucket*-I(0.1), FSC, $GA_{CB}$, and *CPLEX*

| *n* | | 5 | | | 10 | | | 30 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| *m* | | 100 | 250 | 500 | 100 | 250 | 500 | 100 | 250 | 500 |
| *AMMDEM + ls* | ***Gap*** (%) | **0.58** | **0.14** | **0.05** | **0.94** | **0.28** | **0.12** | **1.68** | 0.65 | 0.34 |
| | $\boldsymbol{CPU_{avr}}$ | ***13*** | ***25*** | ***95*** | ***19*** | ***31*** | ***155*** | ***14*** | ***57*** | ***252*** |
| *FSC* | ***Gap*** (%) | **0.58** | 0.15 | **0.05** | **0.94** | 0.31 | 0.13 | **1.68** | 0.66 | **0.33** |
| | $\boldsymbol{CPU_{avr}}$ | 25 | 52 | 205 | 45 | 195 | 872 | 81 | 308 | 1338 |
| *Fixed-bucket-I(0.1)* | ***Gap*** (%) | – | **0.14** | **0.05** | – | **0.28** | **0.12** | – | ***0.64*** | **0.32** |
| | $\boldsymbol{CPU_{avr}}$ | – | 67 | 384 | – | 701 | 1,517 | – | 2,796 | 3,140 |
| $GA_{CB}$ | ***Gap*** (%) | **0.58** | **0.14** | **0.05** | **0.94** | 0.3 | 0.13 | 1.69 | 0.67 | 0.35 |
| | $\boldsymbol{CPU_{avr}}$ | 20 | 174 | 314 | 70 | 276 | 734 | 128.5 | 848 | 1,384 |
| *CPLEX V9* | ***Gap*** (%) | **0.58** | **0.14** | **0.05** | **0.94** | 0.29 | **0.12** | 1.7 | ***0.64*** | 0.33 |
| | $\boldsymbol{CPU_{avr}}$ | 25 | 52 | 205 | 45 | 195 | 872 | 81 | 308 | 1,338 |

Bold underlined italicized results show the best found results. Bold results are the results in which several algorithms found the same result

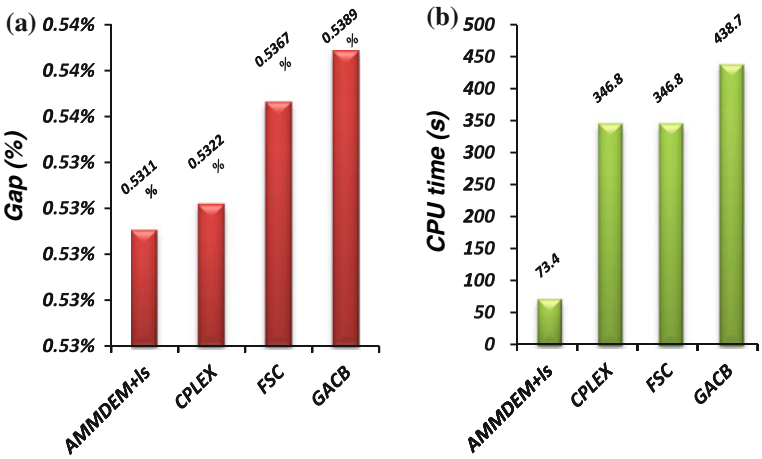


**Fig. 6** The average of (**a**) gap percentage of best found solutions by the methods from LP-relaxation, (**b**) $CPU_{avr}$ (in second) for finding the best solutions over 25 runs when the methods are applied to all 270 problems in OR-Library

standard optimization problems by Birbil and Fang in 2003. Nevertheless, its success was not notable when it tackled discrete space problems such as Travelling Salesman Problem, Nurse Scheduling Problem, etc. In this paper, a modified *SEM* called *DEM* was proposed the operators of which were inspired from the *SEM*'s operators but it could work in the discrete spaces easily. The *SEM* included three main operators (called charge calculation, force calculation, and movement) two of which were modified (force calculation, and movement) and one of which remained unchanged because its original version was consistent with the discrete space

problems. In fact, the subtraction and addition that were extremely used in the *SEM* operators (especially in force calculation and movement operators) were replaced by a specific crossover operator which worked according to the charges of particles. Also, based on the principles of quantum mechanics, a new operator called Annihilation/Creation ($A/A^{\dagger}$) was proposed and applied to the particles according to a probability density function. By applying this operator, a particle is vanished and re-created again in a randomly determined place in the problem space. To show the performance of the method, it was applied to one of the well-studied linear programming problems called Multidimensional Knapsack Problem (*MKP*). To solve the *MKP* using *DEM*, a new effect-based crossover called *MMEUX* (Max–Min Effect-based Uniform Crossover) was proposed. This crossover combined particles according to their charges and reconstructed one new particle. Moreover, it used a max–min approach which bounded the charges of particles and hence increased the exploration of better possible solutions. Furthermore, a probability density function for the $A/A^{\dagger}$ operator was proposed that determined the probability of annihilating a particle according to its history. Three derivations of the *DEM* were established based on the used parameters called *DEM* (which used simple *EUX* crossover without $A/A^{\dagger}$), *MMDEM* (which used *MMEUX* crossover without $A/A^{\dagger}$), and *AMMDEM* (which used *MMEUX* crossover and $A/A^{\dagger}$). After setting the parameters of the method, it was compared with some other population-based methods in terms of the *CPU* time and objective values. Also, the *AMMDEM* + *ls* (*AMMDEM* that has been combined with a local search) was applied to standard test benches and its results were promising. In sum, the proposed method excels several other population-based methods in terms of the *CPU* time (75% improvement in average) and found solutions (1% improvement in average). Some objective values found by the *AMMDEM* + *ls* has been reported in "Appendix".

**Acknowledgments** We would like to express our thanks to the anonymous referees for their valuable advice.

## Appendix

Here, the best found solutions by the *AMMDEM* + *ls* are reported. Due to the results of the test benches 5–100, and 10–100 were solved to optimality (Chu and Beasley 1998), these results are not reported here. It is worthwhile to note that the *AMMDEM* + *ls* could find these values too. Also, the best results of test benches 5–500, 10–500, and 30–500 have been reported in Boussier et al. (2010), Vasquez and Vimont (2005) and are not reported in this appendix. The results of other test benches (5–250, 10–250, 30–100, and 30–250) are reported here. In 10–250 and 30–250 test benches, the *AMMDEM* + *ls* could find optimal solutions as they can be find in Boussier et al. (2010). These values are distinguished by star. Also, the results which were better than $GA_{CB}$ have been ***bolded-italicized*** (Table 8).

**Table 8** The results of 5–250, 10–250, 30–100, and 30–250, found by *AMMDEM* + *ls*

| Problem | Objective | Problem | Objective | Problem | Objective | Problem | Objective |
|---|---|---|---|---|---|---|---|
| **5–250:00** | 59,312 | **10–250:00** | ***59,187**** | **30–100:00** | 21,946 | **30–250:00** | ***56,796*** |
| **5–250:01** | 61,472 | **10–250:01** | 58,672 | **30–100:01** | 21,716 | **30–250:01** | 58,318 |
| **5–250:02** | 62,130 | **10–250:02** | 58,094 | **30–100:02** | 20,754 | **30–250:02** | 56,553 |
| **5–250:03** | 59,446 | **10–250:03** | 61,000* | **30–100:03** | 21,464 | **30–250:03** | ***56,930**** |
| **5–250:04** | 58,951 | **10–250:04** | 58,092* | **30–100:04** | 21,814 | **30–250:04** | 56,629* |
| **5–250:05** | 60,056 | **10–250:05** | 58,803 | **30–100:05** | 22,176 | **30–250:05** | 57,119 |
| **5–250:06** | 60,414 | **10–250:06** | ***58,704**** | **30–100:06** | 21,799 | **30–250:06** | 56,292 |
| **5–250:07** | 61,472 | **10–250:07** | ***58,930*** | **30–100:07** | 21,397 | **30–250:07** | ***56,448*** |
| **5–250:08** | 61,885 | **10–250:08** | ***59,387**** | **30–100:08** | 22,493 | **30–250:08** | 57,442 |
| **5–250:09** | 58,959 | **10–250:09** | ***59,208**** | **30–100:09** | 20,983 | **30–250:09** | 56,447* |
| **5–250:10** | 109,109 | **10–250:10** | ***110,913**** | **30–100:10** | 40,767 | **30–250:10** | ***107,703*** |
| **5–250:11** | 109,841 | **10–250:11** | ***108,669*** | **30–100:11** | 41,304 | **30–250:11** | 108,338 |
| **5–250:12** | 108,489 | **10–250:12** | 108,932* | **30–100:12** | ***41,630*** | **30–250:12** | ***106,409*** |
| **5–250:13** | 109,383 | **10–250:13** | ***110,059*** | **30–100:13** | 41,041 | **30–250:13** | 106,796 |
| **5–250:14** | 110,720 | **10–250:14** | ***108,430*** | **30–100:14** | ***40,889*** | **30–250:14** | ***107,414**** |
| **5–250:15** | 110,256 | **10–250:15** | 110,841 | **30–100:15** | 41,058 | **30–250:15** | 107,246 |
| **5–250:16** | 109,016 | **10–250:16** | 106,075 | **30–100:16** | 41,062 | **30–250:16** | 106,308 |
| **5–250:17** | 109,037 | **10–250:17** | 106,686* | **30–100:17** | 42,719 | **30–250:17** | ***104,003*** |
| **5–250:18** | 109,957 | **10–250:18** | 109,825 | **30–100:18** | 42,230 | **30–250:18** | 106,835 |
| **5–250:19** | 107,038 | **10–250:19** | 106,723* | **30–100:19** | 41,700 | **30–250:19** | 105,751 |
| **5–250:20** | 149,659 | **10–250:20** | ***151,801*** | **30–100:20** | 57,494 | **30–250:20** | ***150,138*** |
| **5–250:21** | 155,940 | **10–250:21** | 148,772* | **30–100:21** | 60,027 | **30–250:21** | 149,907 |
| **5–250:22** | 149,316 | **10–250:22** | 151,900 | **30–100:22** | 58,025 | **30–250:22** | 152,993 |
| **5–250:23** | 152,130 | **10–250:23** | ***151,324**** | **30–100:23** | 60,776 | **30–250:23** | 153,169 |
| **5–250:24** | 150,353 | **10–250:24** | 151,948 | **30–100:24** | 58,884 | **30–250:24** | 150,287* |
| **5–250:25** | 150,045 | **10–250:25** | 152,109* | **30–100:25** | 60,011 | **30–250:25** | 148,544 |
| **5–250:26** | 148,607 | **10–250:26** | 153,131* | **30–100:26** | 58,132 | **30–250:26** | 147,471 |
| **5–250:27** | 149,772 | **10–250:27** | ***153,529*** | **30–100:27** | 59,064 | **30–250:27** | 152,841 |
| **5–250:28** | 155,075 | **10–250:28** | ***149,160**** | **30–100:28** | 58,975 | **30–250:28** | 149,568 |
| **5–250:29** | ***154,668*** | **10–250:29** | 149,704* | **30–100:29** | 60,603 | **30–250:29** | ***149,595*** |

## References


Alaya I, Solnon G, Ghedira K (2004) Ant algorithm for the multidimensional knapsack problem. International conference on bio-inspired optimization methods and their applications. BIOMA 2004, pp 63–72

Alves M, Almeida M (2008) MOTGA: a multiobjective Tchebycheff based genetic algorithm for the multidimensional knapsack problem. Comput Oper Res 34(11):3458–3470

Angelelli E, Mansini R, Speranza MG (2010) Kernel search: a general heuristic for the multi-dimensional knapsack problem. Comput Oper Res 37:2017–2026

Beasley JE (1990) OR-Library: distributing test problems by electronic mail. J Oper Res Soc 41(11):1069–1072

Birbil S, Fang SH (2003) An electromagnetism-like mechanism for global optimization. J Glob Optim, Kluwer, vol 25, pp 263–282

Boussier S, Vasquez M, Vimont Y, Hanafi S, Michelon P (2010) A multi-level search strategy for the 01 Multidimensional Knapsack. Discret Appl Math 158(2):97–109
Chang PC, Chen SH, Fan CY (2009) A hybrid electromagnetism-like algorithm for single machine scheduling problem. Expert Syst Appl 36:1259–1267
Chu P, Beasley JE (1998) A genetic algorithm for the multidimensional knapsack problem. J Heur 4:63–86
Cotta C, Troya JM (1998) "A hybrid genetic algorithm for the 0-1 Multidimensional knapsack problem", Artificial Neural Nets and Genetic Algorithms, Springer, New York, pp 250–254
Debels D, Vanhoucke M (2004) An electromagnetism meta-heuristic for the resource-constrained project scheduling problem. Published in Lecture notes on Computer Science, vol 3871, pp 259–270
Debels D, Vanhoucke M (2006) "The electromagnetism meta-heuristic applied to the resource-constrained project scheduling problem", Lecture notes in Computer Science, ISSN: 0302-9743, pp 259–270
Fidanova S (2002) Evolutionary algorithm for multidimensional knapsack problem. PPSNVII-Workshop
Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. W.H.Freeman and company, New York
Gilmore PC, Gomory RE (1966) The theory and computation of knapsack functions. Oper Res 14:1045–1075
Hanafi S, Fréville A (1998) An efficient tabu search approach for the 0-1 multidimensional Knapsack Problem. Eur J Oper Res 106(2–3):659–675
Hanafi S, Wilbaut C (2008) Scatter search for the 0-1 multidimensional knapsack problem. J Math Model Algor 7:143–159. doi:10.1007/s10852-008-9078-9
Ke L, Feng Z, Ren Z, Wei X (2010) An ant colony optimization approach for the multidimensional knapsack problem. J Heur 16(1):65–83. doi 10.1007/s10732-008-9087-x
Khuri S, Back T, Heitkotter J (1994) The zero/one multidimensional knapsack problem and genetic algorithms. In: Proceedings of the 1994 ACM symposium on applied computing, pp 188–193
Kong M, Tian P (2007) Application of the particle swarm optimization to the multidimensional knapsack problem. Artificial Intelligence and Soft Computing. In: 8th international conference. Proceedings, pp 1140–1149
Kong M, Tian P, Kao Y (2008) A new ant colony optimization algorithm for the multidimensional Knapsack problem. Comput Oper Res 35(8):2672–2683
Leguizamon G, Michalewicz Z (1999) A new version of ant system for subset problems. In: Proceedings of the congress on evolutionary computation, pp 1459–1464
Maenhout B, Vanhoucke M (2007) An electromagnetic meta-heuristic for the nurse scheduling problem. J Heur, Springer, Netherlands, vol 13, no 4, pp 359–385
Martello S, Toth P (1990) Knapsack problems: algorithms and computer implementations. Wiley, New York
Naderi B, Tavakkoli-Moghaddam R, Khalili M (2010) Electromagnetism-like mechanism and simulated annealing algorithms for flowshop scheduling problems minimizing the total weighted tardiness and makespan. Knowl Based Syst 23(2):77–85
Pirkul H (1987) A heuristic solution procedure for the multiconstraint zero-one knapsack problem. Naval Res Logist 34(1):61–72
Rafael PH, Nikitas D (2003) On the performance of the ant colony system for solving the multidimensional Knapsack problem. In: Proceedings of the IEEE pacific rim conference on communications, computers and signal processing, pp 338–341
Shih W (1979) A branch and bound method for the multiconstraint zero-one knapsack problem. J Oper Res Soc 30:369–378
Solnon C, Fenet S (2006) A study of ACO capabilities for solving the maximum clique problem. J Heur 12:155–180
Stenholm S, Suominen KA (2005) Quantum approach to informatics. Wiley, New York
Vasquez M, Vimont Y (2005) Improved results on the 0-1 multidimensional knapsack problem. Eur J Oper Res 165(1):70–81
Wu P, Yang K, Fang H (2006) A revised EM-like algorithm + K-OPT method for solving the traveling salesman problem. In: Proceedings of the first international conference on innovative computing, information and control. ISBN 0-7695-2616-0/06
Yurtkuran A, Emel E (2010) A new hybrid electromagnetism-like algorithm for capacitated vehicle routing problems. Expert Syst Appl 37:3427–3433