Int. J. Mach. Learn. & Cyber.
DOI 10.1007/s13042-014-0272-y

ORIGINAL ARTICLE

Solving 0–1 Knapsack Problem using Cohort Intelligence Algorithm

Anand J. Kulkarni · Hinna Shabir

Received: 26 December 2013 / Accepted: 28 May 2014

Abstract An emerging technique, inspired from the natural and social tendency of individuals to learn from each other referred to as Cohort Intelligence (CI) is presented. Learning here refers to a cohort candidate's effort to self supervise its own behavior and further adapt to the behavior of the other candidate which it intends to follow. This makes every candidate improve/evolve its behavior and eventually the entire cohort behavior. This ability of the approach is tested by solving an NP-hard combinatorial problem such as Knapsack Problem (KP). Several cases of the 0–1 KP are solved. The effect of various parameters on the solution quality has been discussed.The advantages and limitations of the CI methodology are also discussed.

Keywords Cohort Intelligence · Self Supervised Learning · Knapsack Problem · Combinatorial Optimization

1 Introduction

The nature-/bio-inspired optimization techniques such as genetic algorithm (GA), particle swarm optimization (PSO), ant colony optimization (ACO), simulated annealing (SA), tabu search, etc., have become popular due to their simplicity to implement and working based on rules. The GA is population based which is evolved using the operators such as selection, crossover, mutation, etc. The performance of GA is governed by the quality of the population being evaluated and may often reach very close to the global optimal solution and necessitates local improvement techniques to incorporate into it [1, 2]. The paradigm of Swarm Intelligence (SI) is a decentralized self organizing optimization approach inspired from social behavior of living organisms such as insects, fishes, etc. which can communicate with one another either directly or indirectly. The techniques such as Particle Swarm Optimization (PSO) is inspired from the social behavior of bird flocking and school of fish searching for food [3]. The ACO works on the ants' social behavior of foraging food following a shortest path [4]. Similar to ACO, the Bee Algorithm (BA) also works on the social behavior of honey bees finding the food; however, the bee colony tends to optimize the use of number of members involved in particular predecided tasks [5]. The Cuckoo Search (CS) method exhibits slow convergence and marginally low accuracy. In addition, the Improved Cuckoo Search (ICS) uses adaptive step size to adjust search range, and genetic mutation operation to jump out of local optima [6]. Generally, the swarm techniques are computationally intensive. All of these techniques are basically unconstrained optimization methods and their performance can be significantly affected when applied to constrained as well as combinatorial problems.

An emerging Artificial Intelligence (AI) technique referred to as Cohort Intelligence (CI) was proposed in [7]. It is inspired from the self-supervised learning behavior of the candidates in a cohort. The cohort here refers to a group of candidates interacting and competing with one another

A. J. Kulkarni (✉)
Odette School of Business, University of Windsor, 401 Sunset Avenue, Windsor, ON N9B 3P4, Canada
e-mail: kulk0003@ntu.edu.sg; ajkulkarni@oatresearch.org; kulk0003@uwindsor.ca

A. J. Kulkarni · H. Shabir
Optimization and Agent Technology (OAT) Research Lab, Maharashtra Institute of Technology, 124 Paud Road, Kothrud, Pune 411038, India
e-mail: me.hinna@gmail.com

to achieve some individual goal which is inherently common to all the candidates. When working in a cohort, every candidate tries to improve its own behavior by observing the behavior of every other candidate in that cohort. Every candidate may follow a certain behavior in the cohort which according to it may result into improvement in its own behavior. As certain qualities make a particular behavior which, when a candidate follows, it actually tries to adapt to the associated qualities. This makes every candidate learn from one another and helps the overall cohort behavior to evolve. The cohort behavior could be considered saturated, if for considerable number of learning attempts the individual behavior of all the candidates does not improve considerably and candidates' behaviors become hard to distinguish. The cohort could be assumed to become successful when for a considerable number of times the cohort behavior saturates to the same behavior.

The seminal work on CI presented in [7] highlighted that the performance of the CI algorithm was comparable with the PSO and its variations such as the Chaos-PSO (CPSO) [8], Robust Hybrid PSO (RHPSO) [9] and Linearly Decreasing Weight PSO (LDWPSO) [8]. The CPSO searches for better solution in the neighborhood of every particle's current location, the global best solution as well as its local best solution balancing the exploration and exploitation. The RHPSO which is a combination of the PSO with Piecewise Linear Chaotic Map (PWLCM) for increasing the diversity of search further employs Sequential Quadratic Programming (SQP) increasing the efficient search ability in close neighborhood of local minimum. In the LDWPSO the weight associated with the velocity of the particles decreases linearly leading to faster convergence; however, it is essentially governed by the maximum number of iterations for which the algorithm is set to run as well as maximum and minimum allowed weight values. In the collective intelligence algorithm such as PSO [3], the movement of every particle is affected by individual solution as well as the best solution in the swarm. The performance of the overall algorithm is governed by the control parameters such as inertia weight which influence the local and global exploration ability of the algorithm and the acceleration coefficients which influence the quality of the individual and the global solutions. In the CI approach presented here, the collective effort of the swarm is replaced by the competitive nature of cohort candidates in which every candidate tries to follow the behavior of a candidate that has yielded better results in a particular learning attempt. In following this behavior, it tries to incorporate some of the qualities that made that behavior successful. This competitive behavior motivates each candidate to perform better, and leads to an eventual improvement in the individual as well as overall cohort behavior. Similar to one of the variations of the PSO referred to as Barebone PSO (BPSO) [10] in which the best solution is adopted based on the Gaussian distribution, in the CI methodology the roulette wheel approach is used to follow a certain behavior based on certain probability. This approach necessarily helps the CI candidates to jump out of local minima by following worse behavior in few learning attempts. It is also important to mention here that similar to the Fully Informed PSO (FIPSO) [11], in which every particle being fully informed about the other particles in the swarm while the movement is governed by the neighborhood particles; every CI candidate is fully informed of every other candidate's behavior in the entire cohort. Furthermore, similar to the autocatalytic behavior of the ants in the ACO [12, 13] in which the ants follow a random path and as the algorithm progresses, the path being followed the most is favored; in CI the concept of following certain candidate based on some probability which certainly increases with the improvement in its behavior. As mentioned before, due to the probabilistic approach of following a certain behavior, it incorporates ability to jump out of local minima. In addition, when a candidate follows another candidate's behavior, it actually tries to adapt to the associated qualities by exploring in the close neighborhood of the qualities of the candidate being followed. Due to the probabilistic nature of the algorithm few of the candidates may follow the best behavior in the cohort and few may follow worse behavior. This exhibits the balance of both exploration and exploitation property of the algorithm. Recently, CI was modified by introducing a mutation approach [14]. This enlarged searching range and further considerably enhanced the accuracy and convergence speed of the algorithm when applied specifically for the continuous problems only. It was validated by solving several clustering test problems.

The purpose of this paper is to further demonstrate the ability of CI for solving NP-hard combinatorial problem such as the Knapsack Problem (KP). The problem can be divided into two categories, Single-constraint KPs and Multiple-constraint KPs. The single-constraint KPs include the Subset-sum, 0–1 Knapsack, Bounded Knapsack, change-making, and Multiple-choice Knapsack. On the other hand, the multiple-constraint KPs include 0–1 Multiple Knapsack, 0–1 Multidimensional Knapsack, generalized assignment, and Bin Packing with a wide range of applications, such as cargo loading, cutting stock problems, resource allocation in computer systems, and economics [15, 16]. The special case of single constraints is generally known as the KP or the Uni-dimensional KP [17]. Another variant of the KP referred to as Multichoice Multidimensional KP (MMKP) is used to represent an optimally graceful Quality of Service (QoS) degradation model where the QoS of a single session multimedia service is gracefully degraded to conform to changes in resource availability [18]. In [19], the MMKP is used to represent a utility model (UM) which is a mathematical model for a multi-session adaptive multimedia system. The MMKP also appears in the nursing personnel scheduling problem [20], which is defined as the identification of a staffing pattern that specifies the number of nursing personnel of a certain skill to be scheduled and satisfies the total nursing personnel capacity and other relevant constraints. Practical applications of 0–1 Knapsack include finding an optimal investment plan [21], as well as theoretical applications such as a sub-problem when solving generalized assignment problem, which is heavily used when solving vehicle routing problems, efficient packing of cargo containers by considering the weight and volume capacity utilization, etc. [22]. Apart from these applications, KPs are being used for resource allocation problems dealing with the World Wide Web [23]. In this paper various cases of the 0–1 KP [24–26] were solved using CI. In all the problems, the implemented CI methodology produced robust results with reasonable computational cost.

The remainder of this paper is organized as follows. The framework and detailed formulation of the CI methodology is presented in Sect. 2. In Sect. 3, an illustration of applying CI methodology solving the 0–1 KP is presented. Section 4 discusses the results of solving the problems using CI method along with the analysis of various CI parameters. The concluding remarks along with the limitations and associated future directions are discussed in Sect. 5 of the paper. The list of 0–1 KPs are provided in an "Appendix" at the end of the paper.

2 Cohort Intelligence methodology

Consider a general unconstrained problem (in the minimization sense) as follows:

$$\begin{aligned} &\text{Minimize} \quad f(\mathbf{x}) = f(x_1, \ldots x_i, \ldots, x_N) \\ &\text{Subject to} \quad \Psi_i^{lower} \le x_i \le \Psi_i^{upper}, \quad i = 1, \ldots, N \end{aligned} \tag{1}$$

As a general case, assume the objective function $f(\mathbf{x})$ as the behavior of an individual candidate in the cohort which it naturally tries to enrich by modifying the associated set of characteristics/attributes/qualities $\mathbf{x} = (x_1, \ldots x_i, \ldots, x_N)$.

Having considered a cohort with number of candidates C, every individual candidate $c(c = 1, \ldots, C)$ belongs to a set of characteristics/attributes/qualities $\mathbf{x}^c = \left(x_1^c, \ldots x_i^c, \ldots, x_N^c\right)$ which makes the overall quality of its behavior $f(\mathbf{x}^c)$. The individual behavior of each candidate c is generally being observed by itself and every other candidate (c) in the cohort. This naturally motivates every candidate c to follow the behavior better than its current behavior. More specifically, candidate c may follow $f^*\left(\mathbf{x}^{(c)}\right)$ if it is better than $f^*\left(\mathbf{x}^{(c)}\right)$, i.e. $f^*\left(\mathbf{x}^{(c)}\right) < f^*(\mathbf{x}^c)$. Importantly, following a behavior $f(\mathbf{x})$ refers to following associated qualities $\mathbf{x} = (x_1, \ldots x_i, \ldots, x_N)$ with certain variations t associated with them. However, following better behavior and associated qualities is highly uncertain. This is because there is certain probability involved by which it selects certain behavior to follow. In addition, a stage may come where the cohort behavior could become saturated. In other words, at a certain stage, there could be no improvement in the behavior of an individual candidate for a considerable number of learning attempts. Such situation is referred to as saturation stage. This makes every candidate to expand its search around the qualities associated with the current behavior being followed. The mathematical formulation of the CI methodology is explained below in detail with the algorithm flowchart in Fig. 1.

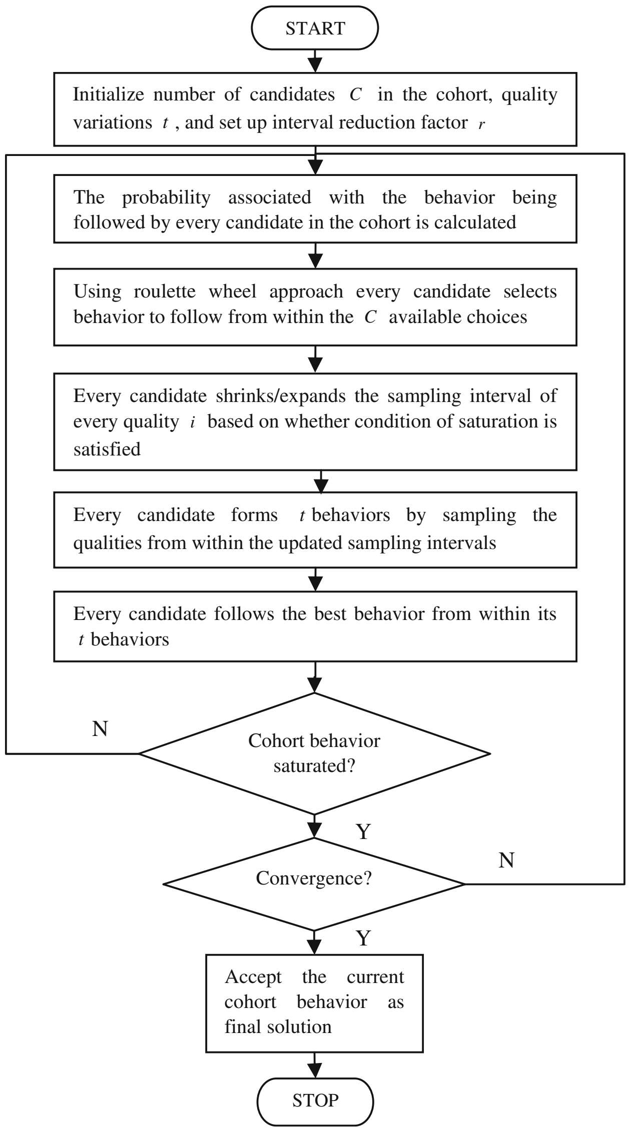
START
Initialize number of candidates C in the cohort, quality variations t, and set up interval reduction factor r
The probability associated with the behavior being followed by every candidate in the cohort is calculated
Using roulette wheel approach every candidate selects behavior to follow from within the C available choices
Every candidate shrinks/expands the sampling interval of every quality i based on whether condition of saturation is satisfied
Every candidate forms t behaviors by sampling the qualities from within the updated sampling intervals
Every candidate follows the best behavior from within its t behaviors
Cohort behavior saturated?
N
Y
Convergence?
N
Y
Accept the current cohort behavior as final solution
STOP

Fig. 1 Cohort Intelligence (CI) flowchart

The procedure begins with the initialization of number of candidates C, sampling interval Ψ_i for each quality x_i, $i = 1, \ldots, N$, learning attempt counter $n = 1$, and the set up of sampling interval reduction factor $r \in [0, 1]$, convergence parameter $\varepsilon = 0.0001$, number of variations t. The values of C, t and v are chosen based on preliminary trials of the algorithm.

Step 1 The probability of selecting the behavior $f^*(\mathbf{x}^c)$ of every associated candidate $c(c = 1, \ldots, C)$ is calculated as follows:

$$p^c = \frac{1/f^*(\mathbf{x}^c)}{\sum_{c=1}^{C} 1/f^*(\mathbf{x}^c)}, \quad (c = 1, \ldots, C) \tag{2}$$

Step 2 Every candidate $c(c = 1, \ldots, C)$ generates a random number $r \in [0, 1]$ and using a *roulette wheel selection* approach decides to follow corresponding behavior $f^*\left(\mathbf{x}^{c[?]}\right)$ and associated qualities $\mathbf{x}^{c[?]} = \left(x_1^{c[?]}, \ldots x_i^{c[?]}, \ldots, x_N^{c[?]}\right)$. The superscript $[?]$ indicates that the behavior is selected by candidate c and not known in advance. The roulette wheel approach could be most appropriate as it provides chance to every behavior in the cohort to get selected purely based on its quality. In addition, it also may increase the chances of any candidate to select the better behavior as the associated probability stake $p^c(c = 1, \ldots, C)$ presented in Eq. (2) in the interval [0, 1] is directly proportional to the quality of the behavior $f^*(\mathbf{x}^c)$. In other words, better the solution, higher is the probability of being followed by the candidates in the cohort.

Step 3 Every candidate $c(c = 1, \ldots, C)$ shrinks the sampling interval $\Psi_i^{c[?]}, \quad i = 1, \ldots, N$ associated with every variable $x_i^{c[?]}, \quad i = 1, \ldots, N$ to its local neighborhood. This is done as follows:

$$\Psi_i^{c[?]} \in \left[x_i^{c[?]} - (\|\Psi_i\|/2),\ x_i^{c[?]} + (\|\Psi_i\|/2)\right] \tag{3}$$

where $\Psi_i = (\|\Psi_i\|) \times r$.

Step 4 Each candidate $c(c = 1, \ldots, C)$ samples t qualities from within the updated sampling interval $\Psi_i^{c[?]}, \quad i = 1, \ldots, N$ associated with every variable $x_i^{c[?]}, \quad i = 1, \ldots, N$ and computes a set of associated t behaviors, i.e. $\mathbf{F}^{c,t} = \left\{f(\mathbf{x}^c)^1, \ldots, f(\mathbf{x}^c)^j, \ldots, f(\mathbf{x}^c)^t\right\}$, and selects the best function $f^*(\mathbf{x}^c)$ from within. This makes the cohort is available with C updated behaviors represented as $\mathbf{F}^C = \left\{f^*(\mathbf{x}^1), \ldots, f^*(\mathbf{x}^c), \ldots, f^*(\mathbf{x}^C)\right\}$.

Step 5 The cohort behavior could be considered saturated, if there is no significant improvement in the behavior $f^*(\mathbf{x}^c)$ of every candidate $c(c = 1, \ldots, C)$ in the cohort, and the difference between the individual behaviors is not very significant for successive considerable number of learning attempts, i.e. if

1. $\left\|\max\left(\mathbf{F}^C\right)^n - \max\left(\mathbf{F}^C\right)^{n-1}\right\| \le \varepsilon$, and
2. $\left\|\min\left(\mathbf{F}^C\right)^n - \min\left(\mathbf{F}^C\right)^{n-1}\right\| \le \varepsilon$, and
3. $\left\|\max\left(\mathbf{F}^C\right)^n - \min\left(\mathbf{F}^C\right)^n\right\| \le \varepsilon$, every candidate $c(c = 1, \ldots, C)$ expands the sampling interval $\Psi_i^{c[?]}, \ i = 1, \ldots, N$ associated with every quality $x_i^{c[?]}, \ i = 1, \ldots, N$ to its original one $\Psi_i^{lower} \le x_i \le \Psi_i^{upper}, \ i = 1, \ldots, N$.

Step 6 If either of the two criteria listed below is valid, accept any of the C behaviors from current set of behaviors in the cohort as the final objective function value $f^*(\mathbf{x})$ as the final solution and stop, else continue to Step 1.

(a) If maximum number of attempts exceeded.
(b) If cohort saturates to the same behavior (satisfying the conditions in Step 5) for $\tau_{\max}$ times.

3 0–1 Knapsack Problem using CI method

The KP is a well-known NP-Hard combinatorial problem described as follows [24–26]: given a set of N objects, each object $i, \quad i = 1, \ldots, N$ is associated with an integer profit v_i and an integer weight w_i. Fill the knapsack with a subset of the objects such that the total profit $f(\mathbf{v})$ is maximized and the total weight $f(\mathbf{w})$ does not exceed a given capacity W. The mathematical formulation is as follows:

$$\text{Maximize } f(\mathbf{v}) = \sum_{i=1}^{N} v_i x_i$$

$$\text{Subject to } f(\mathbf{w}) \le W$$

where

$$f(\mathbf{w}) = \sum_{i=1}^{N} w_i x_i, \ x_i \in \{0, 1\}, \ 1 \le i \le N \tag{4}$$

3.1 Illustration of CI solving 0–1 KP

In the context of CI algorithm (discussed in Sect. 2), the objects $i, \quad i = 1, \ldots, N$are considered as characteristics/attributes/qualities which decide the overall profit $f(\mathbf{v})$ and the associated overall weight $f(\mathbf{w})$of the knapsack. The procedure begins with the initialization of the number of cohort candidates C, and the number of variations t. In the cohort of C candidates, initially every candidate $c \ (c = 1, \ldots, C)$ randomly selects few objects, and the associated profits $\mathbf{F}^C = \left\{f(\mathbf{v}^1), \ldots, f(\mathbf{v}^c), \ldots, f(\mathbf{v}^C)\right\}$ and weights $\mathbf{F}^{CW} = \left\{f(\mathbf{w}^1), \ldots, f(\mathbf{w}^c), \ldots, f(\mathbf{w}^C)\right\}$ are calculated. The further CI algorithm steps are discussed below.

Step 1 The probability p^c $(c = 1,\ldots,C)$ of selecting a profit $f(\mathbf{v}^c)$, $(c = 1,\ldots,C)$, is calculated as $p^c = p_1^c + p_2^c$ where

$$p_1^c = \frac{f(\mathbf{v}^c)}{\sum_{c=1}^{C} f(\mathbf{v}^c)} \tag{5}$$

and

$$p_2^c = \begin{cases} \dfrac{f(\mathbf{w}^c)}{W}, & f(\mathbf{w}^c) \le W \\ 3 - \dfrac{2f(\mathbf{w}^c)}{W}, & f(\mathbf{w}^c) > W \end{cases} \tag{6}$$

A probability distribution specially devised to bias the solution towards feasibility is represented in Fig. 2. The probability p_2^c increases linearly as the total weight of the knapsack increases, and reaches its peak value at the maximum capacity W. Upon any further increase in weight the probability rapidly decreases. Thus, the probability is highest around maximum capacity and decreases on either side of it with decrease beyond W with twice the slope.

Step 2 Based on roulette wheel selection approach every candidate c $(c = 1,\ldots,C)$ selects a candidate with associated profit $f(\mathbf{v}^{c[?]})$ and modifies its own solution by incorporating some objects from that candidate. The superscript [?] indicates that the behavior is selected by candidate c and not known in advance. The modification approach is inspired from the feasibility-based rules discussed in [29–31]. The modifications are categorized as follows:

1. If the solution of candidate c $(c = 1,\ldots,C)$ is feasible, i.e. it satisfies the weight constraint given by Eq. (4) then, it randomly chooses one of the following modifications:

   1.1. Adds a randomly chosen object from the candidate being followed, such that the object has not been included in the present candidate c and the weight constraint given by Eq. (4) is still satisfied.

   1.2. Replaces a randomly chosen object with another randomly chosen one from the candidate being followed, such that Eq. (4) is satisfied

2. If the candidate c $(c = 1,\ldots,C)$ is infeasible then, it randomly chooses one of the following modifications:

   2.1. Removes a randomly chosen object from within its knapsack.

   2.2. Replaces a randomly chosen object with another randomly chosen one from the candidate c being followed, such that the total weight $f(\mathbf{w}_c)$ of the candidate c decreases.

   Every candidate performs the above procedure t times. This makes every candidate c available with associated profits $\mathbf{F}^{c,t} = \left\{ f(\mathbf{v}^c)^1,\ldots,f(\mathbf{v}^c)^j,\ldots,f(\mathbf{v}^c)^t \right\}$, $(c = 1,\ldots,C)$. Furthermore, every candidate selects the best profit $f^*(\mathbf{v})$ among them. The best variation is selected based on the following conditions:

   2.2.1. If the variations are feasible then the variation with maximum profit is selected.

   2.2.2. If the variations are infeasible then the variation with minimum weight is selected.

   2.2.3. If there are both infeasible and feasible variations then the feasible variation with maximum profit is selected.

This makes the cohort available with C updated profits $\mathbf{F}^C = \left\{ f^*(\mathbf{v}^1),\ldots,f^*(\mathbf{v}^c),\ldots,f^*(\mathbf{v}^C) \right\}$.

This process continues until saturation (convergence) i.e., every candidate has the same profit and it does not change for successive considerable number of learning attempts.

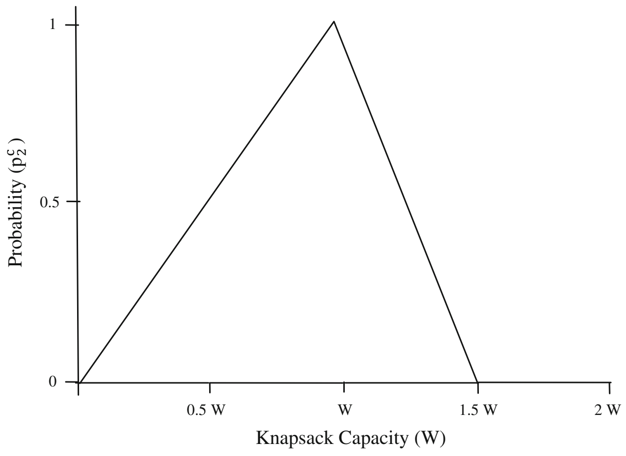

Fig. 2 Probability distribution for p_2^c

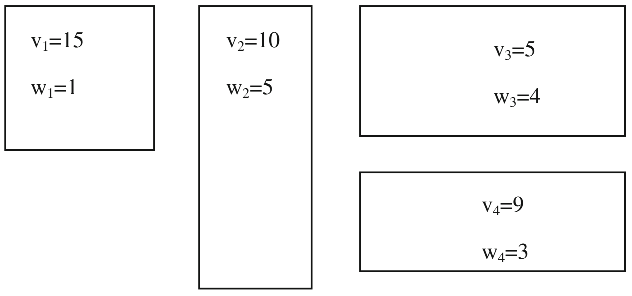

Fig. 3 Illustrative 0–1 KP example with $N = 4$, $W = 8$

The above discussed procedure of solving the KP using CI algorithm is illustrated here with number of objects $N = 4$ and knapsack capacity $W = 8$. The weights $w_i, \ i = 1,\ldots,N$ and profits $v_i, \ i = 1,\ldots,N$ associated with every object are illustrated in Fig. 3. Furthermore, the cohort is assumed to have three candidates, i.e. $C = 3$ and number of variations $t = 3$.

Initially every candidate $c(c = 1, \ldots, C)$ randomly selects few objects, and the associated profits $\mathbf{F}^C = \{f(\mathbf{v}^1),\ldots,f(\mathbf{v}^c),\ldots,f(\mathbf{v}^C)\}$ and weights $\mathbf{F}^{C,W} = \{f(\mathbf{w}^1),\ldots,f(\mathbf{w}^c),\ldots,f(\mathbf{w}^C)\}$ are calculated. The further CI algorithm steps are discussed below:

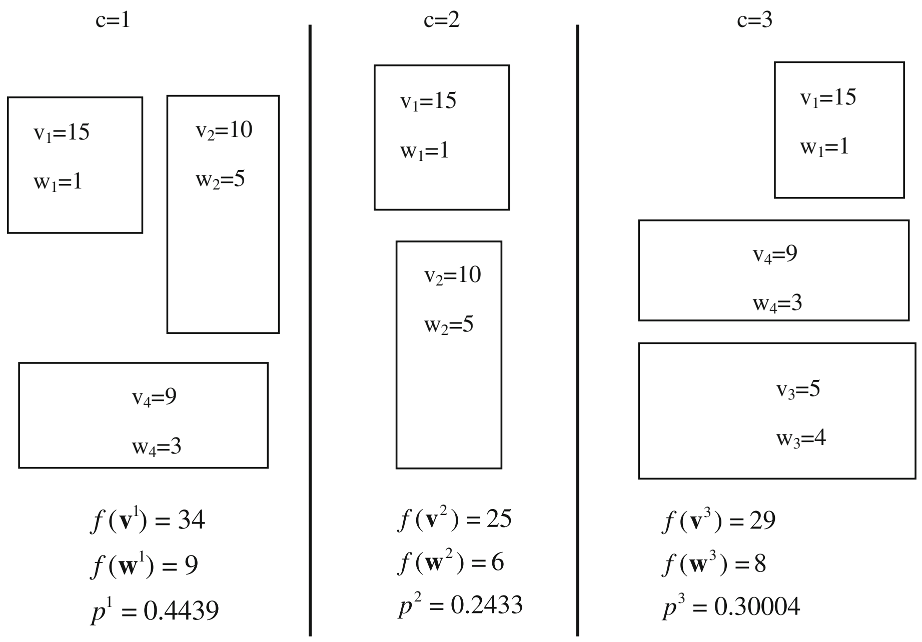

Fig. 4 Illustrative 0–1 KP example with $C = 3$

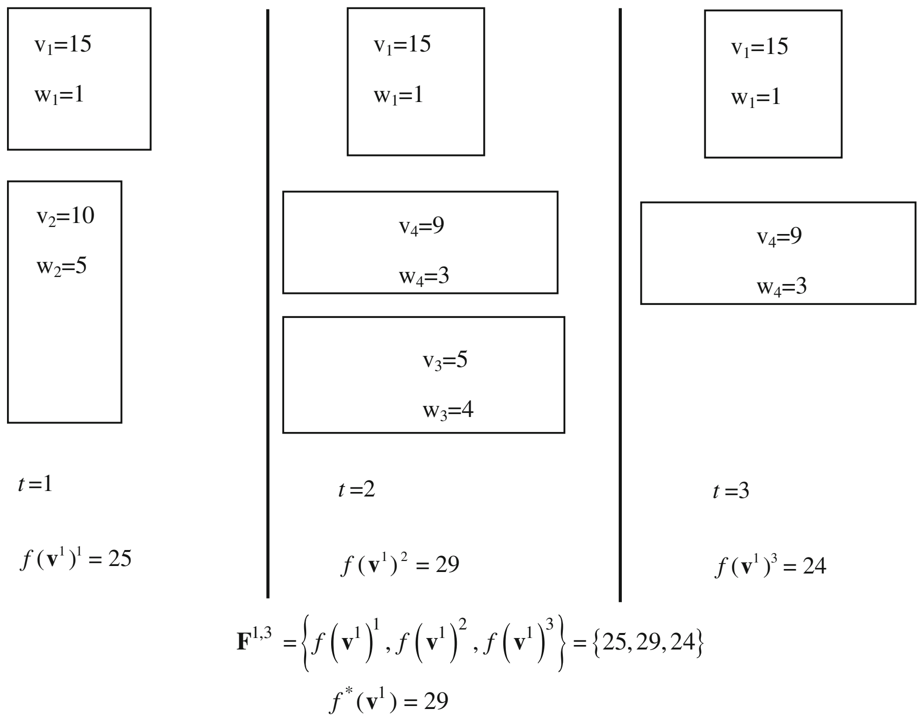

Fig. 5 Illustrative 0–1 KP Example with $t = 3$(variations obtained)

1. The probability p^c associated with each candidate $c(c = 1, \ldots, 3)$ is calculated using Eq. (5) and (6). The calculated probability values are presented in Fig. 4.
2. Using roulette wheel selection approach, assume that candidate 1 decides to follow candidate 3. As presented in Fig. 5, $t = 3$ variations are formed along with the associated profits vector $\mathbf{F}^{1,3} = \left\{f(\mathbf{v}^1)^1, f(\mathbf{v}^1)^2, f(\mathbf{v}^1)^3\right\}$ and the selected variation with profit $f^*(\mathbf{v}^1)$. In this way, candidates 2 and 3 also follow certain candidate and update themselves. It makes the cohort available with 3 updated candidates with profits $\mathbf{F}^3 = \left\{f^*(\mathbf{v}^1), f^*(\mathbf{v}^2), f^*(\mathbf{v}^3)\right\}$. This process continues until saturation (convergence), i.e. every candidate finds the solution and does not change for successive considerable number of learning attempts.

As $c = 1$ is infeasible it can modify itself by either removing an object or replacing one. The variations formed by it are: $t = 1$: remove an object from c = 1 profit of candidate $f(\mathbf{v}^1)^1$: 25; $t = 2$: replace an object in c = 1 with an object from c = 3 profit of candidate: $f(\mathbf{v}^1)^2$: 29; $t = 3$: remove an object from c = 1, Profit of candidate: $f(\mathbf{v}^1)^3$: 24.

4 Results and discussion

The CI algorithm discussed in Sect. 2 was coded in Matlab 7.11 (R2010b) and the simulations were run on a Windows platform using an Intel Core i5 CPU, 2.27 GHz processor speed and 3 GB memory capacity, and further validated using twenty distinct test cases of the 0–1 Knapsack Problems. The standard test cases $f_1 - f_{10}$ [24–26] are presented in Table 1. The cases $f_{11} - f_{20}$ were generated using a random number generator. In these tests, knapsack capacity is calculated using the formula [25, 26]: $W = \frac{3}{4}\sum_{i=1}^{N} w_i$ where w_i is a random weight of item i and N is the number of items. Different values of N were used, varying from 30 to 75. These test cases are presented in an "Appendix" at the end of the paper.

Recently, the instances $f_1 - f_{10}$ were solved using Harmony Search (HS) [24, 27], Improved Harmony Search (IHS) [24, 28], Novel Global Harmony Search (NGHS)

Table 1 The dimension and parameters of ten test problems

f	Number of objects (N)	Parameters (W, w, v)
f_1	10	$W = 269$, $\mathbf{w} = \{95, 4, 60, 32, 23, 72, 80, 62, 65, 46\}$, $\mathbf{v} = \{55, 10, 47, 5, 4, 50, 8, 61, 85, 87\}$
f_2	20	$W = 878$, $\mathbf{w} = \{92, 4, 43, 83, 84, 68, 92, 82, 6, 44, 32, 18, 56, 83, 25, 96, 70, 48, 14, 58\}$, $\mathbf{v} = \{44, 46, 90, 72, 91, 40, 75, 35, 8, 54, 78, 40, 77, 15, 61, 17, 75, 29, 75, 63\}$
f_3	4	$W = 20$, $\mathbf{w} = \{6, 5, 9, 7\}$, $\mathbf{v} = \{9, 11, 13, 15\}$
f_4	4	$W = 11$, $\mathbf{w} = \{2, 4, 6, 7\}$, $\mathbf{v} = \{6, 10, 12, 13\}$
f_5	15	$W = 375$, $\mathbf{w} = \{56.358531, 80.874050, 47.987304, 89.596240, 74.660482, 85.894345, 51.353496, 1.498459, 36.445204, 16.589862, 44.569231, 0.466933, 37.788018, 57.118442, 60.716575,\}$ $\mathbf{v} = \{0.125126, 19.330424, 58.500931, 35.029145, 82.284005, 17.410810, 71.050142, 30.399487, 9.140294, 14.731285, 98.852504, 11.908322, 0.891140, 53.166295, 60.176397\}$
f_6	10	$W = 60$, $\mathbf{w} = \{30, 25, 20, 18, 17, 11, 5, 2, 1, 1\}$, $\mathbf{v} = \{20, 18, 17, 15, 15, 10, 5, 3, 1, 1\}$
f_7	7	$W = 50$, $\mathbf{w} = \{31, 10, 20, 19, 4, 3, 6\}$, $\mathbf{v} = \{70, 20, 39, 37, 7, 5, 10\}$
f_8	23	$W = 10000$, $\mathbf{w} = \{983, 982, 981, 980, 979, 978, 488, 976, 972, 486, 486, 972, 972, 485, 485, 969, 966, 483, 964, 963, 961, 958, 959,\}$ $\mathbf{v} = \{981, 980, 979, 978, 977, 976, 487, 974, 970, 485, 485, 970, 970, 484, 484, 976, 974, 482, 962, 961, 959, 958, 857\}$
f_9	5	$W = 80$, $\mathbf{w} = \{15, 20, 17, 8, 31\}$, $\mathbf{v} = \{33, 24, 36, 37, 12\}$
f_{10}	20	$W = 879$, $\mathbf{w} = \{84, 83, 43, 4, 44, 6, 82, 92, 25, 83, 56, 18, 58, 14, 48, 70, 96, 32, 68, 92,\}$ $\mathbf{v} = \{91, 72, 90, 46, 55, 8, 35, 75, 61, 15, 77, 40, 63, 75, 29, 75, 17, 78, 40, 44\}$

[24–26], Quantum Inspired Cuckoo Search Algorithm QICSA [26], and Quantum Inspired Harmony Search Algorithm (QIHSA) [25].The HS is based on natural musical performance processes and has been applied to a variety of engineering problems; however, it exhibits poor convergence rate [24]. IHS employs a parameter updating method for generating new solution vectors that enhances accuracy and convergence rate of HS algorithm. The convergence rate is further improved in NGHS which is inspired from the swarm intelligence and employs a dynamic updating strategy and probabilistic mutation approach; however, the performance degenerates significantly when applied for solving constrained problems. All these algorithms lack a method to satisfy constraints and hence, can result in an infeasible solution when solving constrained optimization problems. In [24] a penalty function method is used along with NGHS in order to handle the weight constraint in 0–1 KP. QICSA integrates the quantum computing principles such as qubit representation, measure operation and quantum mutation, in the Cuckoo Search algorithm. It is different from other evolutionary algorithms in that it offers a large exploration of the search space through intensification and diversification [26]. QIHSA combines the features of HS algorithm and quantum computing. The probabilistic nature of the quantum measure offers a good diversity to the harmony search algorithm, while the interference operation helps to intensify the search around the best solutions [25]. While hybridization between quantum inspired computing and nature inspired algorithms significantly improve the performance over the original nature inspired algorithms, their performance depends largely on the initial solutions, which are selected randomly. Also, when dealing with constrained optimization problems they require the use of a repair operator.

The approach of CI handles constraints using a probability distribution p_2^c (refer to Fig. 2) which forces the candidates to follow the behaviour/solution with constraints satisfied as well as closer to the ones with constraint values closer to the boundary. Moreover, a well-established feasibility-based rule [29–31] was also incorporated which assists candidates select the variations with better objective and constraint satisfaction (refer to Sect. 3.1). The summary of the CI results including the best, mean and worst solutions with the associated average CPU time, average number of function evaluations, standard deviation are listed in Table 2. In addition, the CI parameters such as number of candidates C and number of

Table 2 Summary of solutions of KPs solved using CI

Problem	Number of objects, Knapsack capacity (N, W)	Solution (f^*(v), f^*(w)) Best	Mean	Worst	Standard deviation	Average function evaluations (FE)	Average time (s)	Parameters (C, t)
f_1	10, 269	295, 269	267.46, 262.722	260, 250	0.0	5,410	0.4489	5, 10
f_2	20, 878	1,024, 871	1,020.55, 852.84	1,009, 827	0.0	5,446	1.5909	5, 10
f_3	4, 20	35, 18	34.55, 17.867	28, 16	0.0	5,136	0.2687	5, 10
f_4	4, 11	23, 11	22.06, 10.33	16, 6	0.64	5,193	0.2492	5, 10
f_5	15, 375	481.0694, 354.9608	449.986, 361.692	412.6988, 372.9118	10.68	5,590	0.6609	5, 10
f_6	10, 60	51, 56	50.733, 56.733	49, 54	0.66	5,573	0.4465	5, 10
f_7	7, 50	105, 50	86.6, 44.8	79, 42	2.99	5,696	0.3749	5, 10
f_8	23, 10,000	9,759, 9,760	9,753.33, 9,756.33	9,710, 9,711	11.5	6,486	1.1959	5, 10
f_9	5, 80	130, 60	124.6, 61.4	106, 74	2.89	5,110	0.3048	5, 10
f_{10}	20, 879	1,025, 871	997.7, 558.3	892, 805	18.6	5,426	1.535	5, 10
f_{11}	30, 577	1,437, 566	1,418, 571.5	1,398, 563	11.79	6,817	3.4635	5, 10
f_{12}	35, 655	1,689, 650	1,686.5, 650.833	1,679, 654	3.8188	5,375	5.2288	5, 10
f_{13}	40, 819	1,816, 817	1,807.5, 817.66	1,791, 819	9.604	7,833	7.3429	5, 10
f_{14}	45, 907	2,020, 903	2,017, 902.5	2,007, 901	4.749	7,433	8.1510	5, 10
f_{15}	50, 882	2,440, 873	2,436.166, 870.33	2,421, 865	6.841	7,766	10.5690	5, 10
f_{16}	55, 1,050	2,643, 1,049	2,605, 1,047.8	2,581, 1,049	22.018	9,720	14.3445	5, 10
f_{17}	60, 1,006	2,917, 1,002	2,915, 1,001.833	2,905, 1,001	4.472	9,017	17.0894	5, 10
f_{18}	65, 1,319	2,814, 1,319	2,773.66, 1,316.33	2,716, 1,317	18.273	10,283	20.9486	5, 10
f_{19}	70, 1,426	3,221, 1,426	3,216, 1,423.166	3,211, 1,419	4.3589	10,333	26.4846	5, 10
f_{20}	75, 1,433	3,614, 1,432	3,603.8, 1,431.8	3,591, 1,429	8.035	12,720	34.0072	5, 10

Table 3 Comparison of results obtained using CI with other established methods

Problem	Number of objects (N)	Method	Optimum solution $f^*(\mathrm{v})$
f_1	10	HS [24]	295
		IHS [24]	295
		NGHS [24, 25]	295
		QICSA [25]	295
		QIHSA [26]	295
		CI	295
f_2	20	HS [24]	1,024
		IHS [24]	1,024
		NGHS [24, 25]	1,024
		QICSA [25]	1,024
		QIHSA [26]	1,024
		CI	1,024
f_3	4	HS [24]	35
		IHS [24]	35
		NGHS [24, 25]	35
		QICSA [25]	35
		QIHSA [26]	35
		CI	35
f_4	4	HS [24]	23
		IHS [24]	23
		NGHS [24, 25]	23
		QICSA [25]	23
		QIHSA [26]	23
		CI	23
f_5	15	HS [24]	481.0694
		IHS [24]	481.0694
		NGHS [24, 25]	481.0694
		QICSA [25]	481.0694
		QIHSA [26]	481.0694
		CI	481.0694
f_6	10	HS [24]	50
		IHS [24]	50
		NGHS [24, 25]	52
		QICSA [25]	52
		QIHSA [26]	52
		CI	51
f_7	7	HS [24]	107
		IHS [24]	107
		NGHS [24, 25]	107
		QICSA [25]	107
		QIHSA [26]	107
		CI	105
f_8	23	HS [24]	9,767
		IHS [24]	9,767
		NGHS [24, 25]	9,767
		QICSA [25]	9,767
		QIHSA [26]	9,767
		CI	9,759
f_9	5	HS [24]	130
		IHS [24]	130
		NGHS [24, 25]	130
		QICSA [25]	130
		QIHSA [26]	130
		CI	130
f_{10}	20	HS [24]	1,025
		IHS [24]	1,025
		NGHS [24, 25]	1,025
		QICSA [25]	1,025
		QIHSA [26]	1,025
		CI	1,025

variations t are also listed. As presented in Table 3, it can be seen that the solution was comparable for all problems and in most of the cases the optimum solution was obtained. In addition, according to Table 2, it is clear that the solution was obtained in reasonable computational cost (time and FE). The results have also been verified with Branch and Bound method, and according to Tables 3, 4 and Fig. 6 it is clear that the performance of CI and Branch and Bound are quite comparable. The CI saturation/convergence plot for one of the problems, $f_{10}(N = 20)$ is presented in Fig. 7 which illustrates the self adaptive learning behavior of every candidate in the cohort. Initially, the distinct behavior of every individual candidate in the cohort can be easily distinguished. As every candidate adopts the qualities of other candidates to improve its own solution, the cohort saturates to a certain improved solution. It is noted that the standard deviation was quite narrow with smaller sized problems; however, increased as the problem size increased. In addition, computational cost, i.e. time and function evaluations also increased with increase in the problem size. However, it was observed that in few runs of CI the candidates converged at suboptimal solutions. Similar to the perturbation approach implemented in [31], in order to make the candidates jump out of possible local minima, every candidate $c\ (c = 1, \ldots, C)$ randomly selects a candidate to follow without considering its effect on the solution. This approach instantaneously made the solution worse, however, it was found to be helpful to pull the candidates' solution out of local minima and reach an improved solution. This approach was much simpler as opposed to the perturbation approach in [31] where several

Table 4 Comparison of results obtained using CI with B&B

Problem	Number of objects (N)	Optimum solution (f^*(v)) CI B&B	Time (s)
f_1	10	295	0.4489
		295	0.12
f_2	20	1,024	1.5909
		1,024	0.04
f_3	4	35	0.2687
		35	0.03
f_4	4	23	0.2492
		23	0.03
f_5	15	481.0694	0.6609
		481.0690	0.18
f_6	10	51	0.4465
		52	0.14
f_7	7	105	0.3749
		107	0.04
f_8	23	9,759	1.1959
		9,767	0.18
f_9	5	130	0.3048
		130	0.03
f_{10}	20	1,025	1.535
		1,025	0.45
f_{11}	30	1,437	3.4635
		1,437	0.156001
f_{12}	35	1,689	5.2288
		1,689	0.0624004
f_{13}	40	1,816	7.3429
		1,821	0.0156001
f_{14}	45	2,020	8.1510
		2,033	0.0312002
f_{15}	50	2,440	10.5690
		2,440	0.0312002
f_{16}	55	2,643	14.3445
		2,440	0.0312002
f_{17}	60	2,917	17.0894
		2,917	0.0312002
f_{18}	65	2,814	20.9486
		2,818	0.0624004
f_{19}	70	3,221	26.4846
		3,223	0.0780005
f_{20}	75	3,614	34.0072
		3,614	0.0312002

parameters were required to be tuned based on the preliminary trials.

The effect of CI parameters viz. the number of candidates $C$and the number of variations in behaviour t was analyzed using the final values of profit $f(\mathbf{v})^*$, the total number of function evaluations and the computational time, for each problem. For every pair of number of candidates C and the number of variations in behavior t every KP test case was solved 20 times. For all the problems, the computational cost, i.e. the number of function evaluations and computational time was observed to be increasing with increasing number of candidates C, as well as number of variations in behaviour t. This was because, with increase in the number of candidates C and variations t, the number of behavior choices i.e. number of function evaluations also increased. The average values of profit $f(\mathbf{v})^*$, the total number of function evaluations and the computational time for different values of number of candidates C and variations t are shown for problem f_1 in Figs. 8, 9 and 10, respectively. Another important observation was that as the problem size i.e. N increased, the computational cost also increased (refer to Table 2). Therefore, problems with larger number of objects took a longer time and more number of function evaluations to converge. Furthermore, with fewer number of candidates C, the solution, i.e. total profit $f(\mathbf{v})^*$ was suboptimal. As the value of number of candidates C was increased the solution quality improved up to a certain point after which there was no significant change (refer to Fig. 8). This was because for small values of number of candidates$C$the behavior choices were few and as number of candidates C increased the behavior choices increased and hence, the chances of selecting a better solution increased. In most of the problems there wasn't any significant change in the solution beyond $C = 5$. At the same time for some problems such as f_1 with small values of problem size N the optimum solution was reached at $C = 3$ and no significant change was observed in the solution upon further increase in number of candidates C. Thus, the effect of number of candidates C on the solution was dependent on the problem size N. In addition, it was observed that even with large values of number of candidates C the solution was suboptimal if the number of variations t was small. As the value of t increased, the solution quality improved. For most of the problems no significant change was observed in the solution beyond $t = 10$. For some problems such as f_1, with small values of problem size N optimum solution was obtained at $t = 4$ and no significant change was seen in the solution upon further increasing the number of variations t. Therefore, even in case of the number of variations t, its effect on the solution was dependent on the problem size N. Accordingly, for all problems the number of candidates C and number of variations $t$were chosen to be 5 and 10, respectively.

5 Conclusions and future directions

For the first time emerging CI algorithm has been applied for solving a combinatorial NP-hard problem such as 0–1 KP, with number of objects varying from 4 to 75. In all the

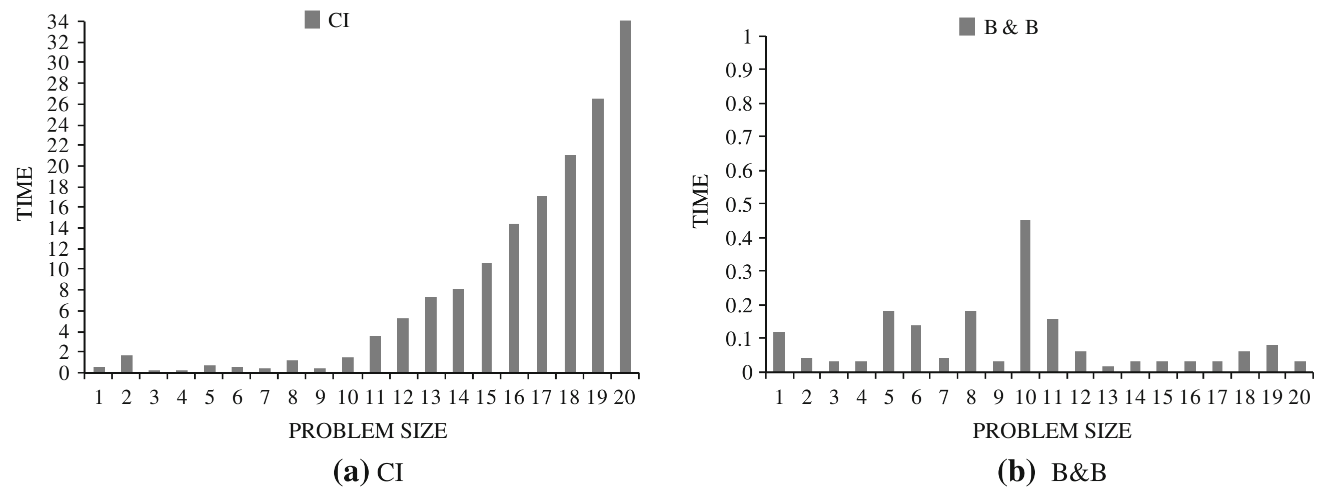

Fig. 6 Effect of problem size (*N*) on computational time

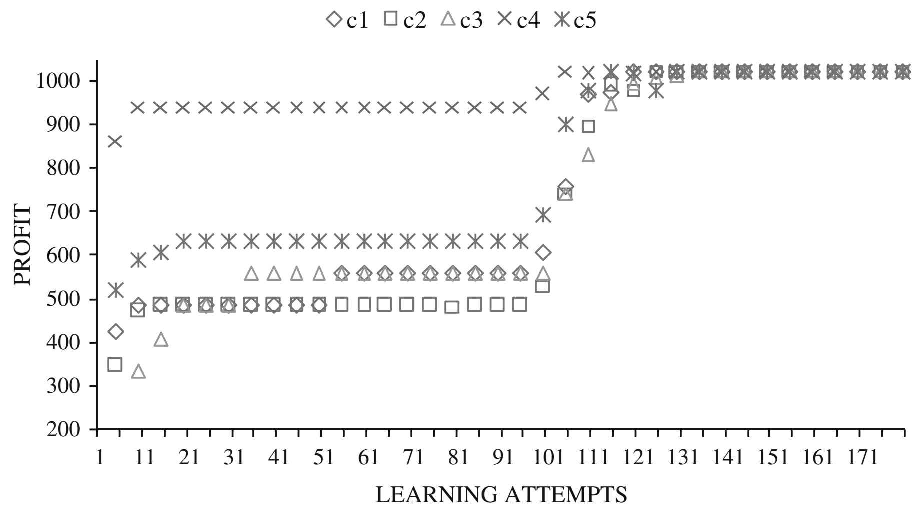

Fig. 7 Learning attempts vs. the objective function $f(v)^*$ for each candidate

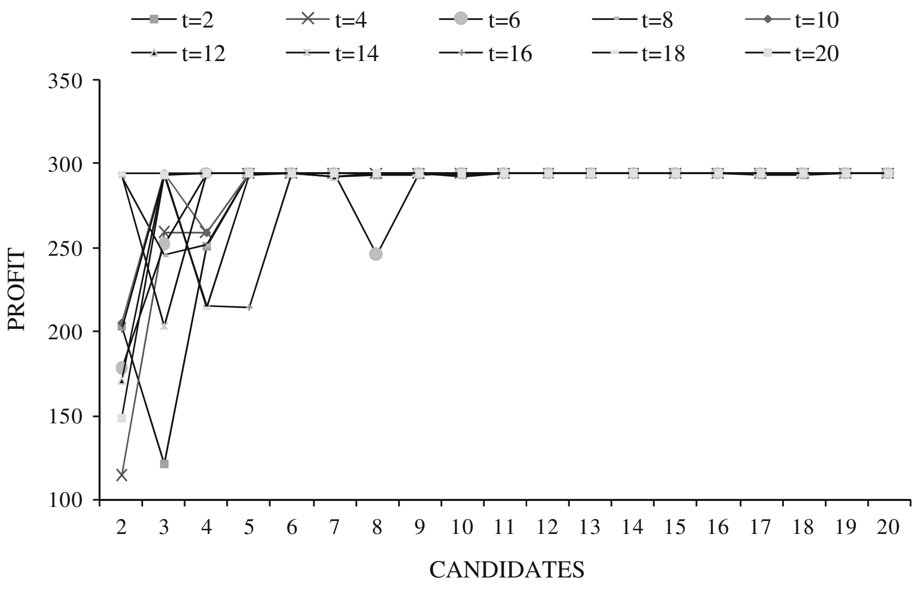

Fig. 8 Effect of number of candidates (*C*) on the profit $f(\mathbf{v})^*$ for different values of variations (*t*)

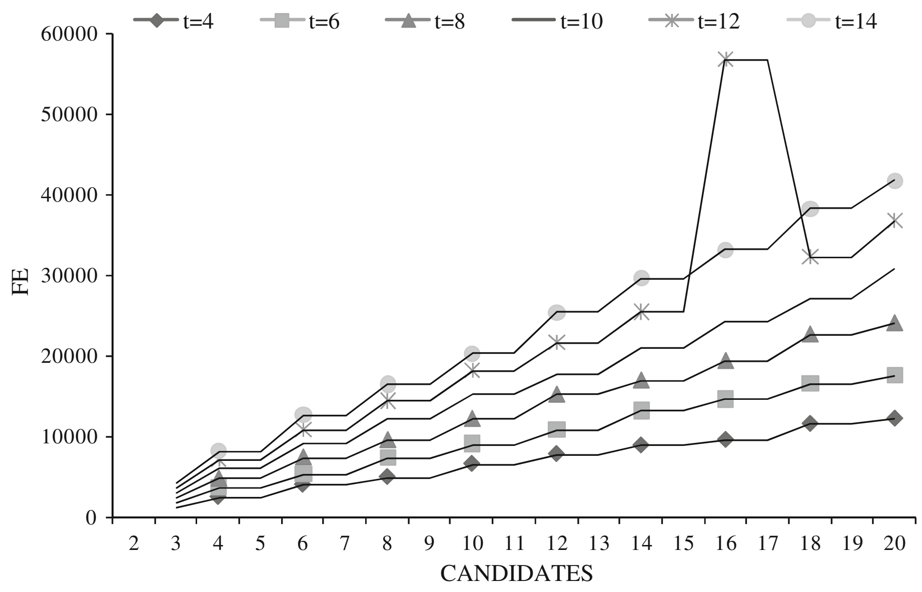

Fig. 9 Effect of number of candidates (C) on the function evaluations (FE) for different values of variations (t)

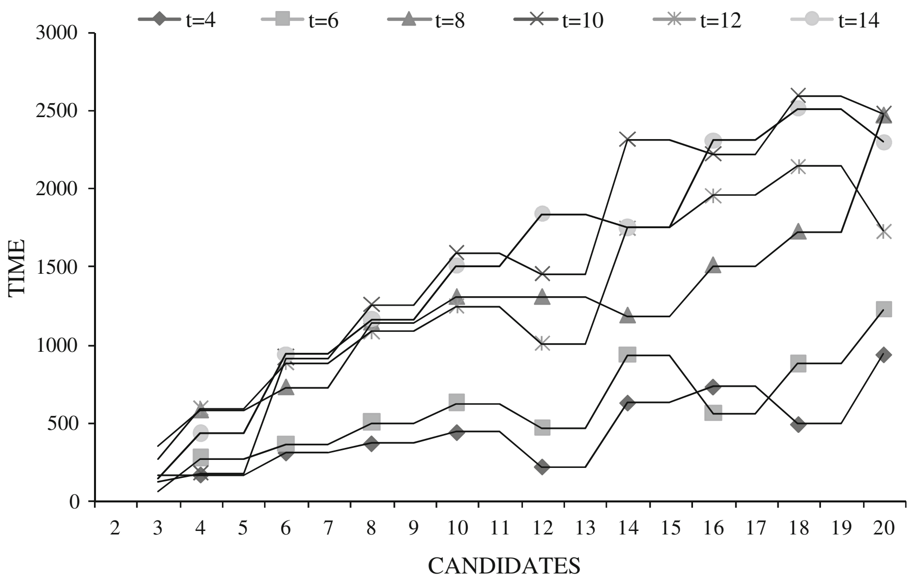

Fig. 10 Effect of number of candidates (C) on the time for different values of variations (t)

problems the implemented CI methodology produced satisfactory results with reasonable computational cost. Furthermore, according to the solution comparison of CI with other contemporary methods it could be seen that the CI solution is comparable and for some problems even better than the other methods. The CI methodology was therefore validated and the self supervising nature of the cohort candidates was successfully demonstrated along with their ability to learn and improve qualities which further improved their individual behavior. In addition, in order to avoid saturation of cohort at suboptimal solution and further make the cohort saturate to the optimum solution, a generic approach such as accepting random behavior was incorporated. The effect of the important parameters such as number of candidates C and the associated variations $t$on the computational time, function evaluations and the solution was analysed. This could be a useful reference in dealing with future problems using CI.

As an emerging technique, CI is in its early stage of development. It was observed that the computational time and function evaluations of the CI algorithm increased considerably with the problem size, in the future a self-adaptive scheme could be developed for these parameters such as number of candidates $C$and number of variations t. This may make CI algorithm computationally more efficient and improve the rate of convergence. In addition, authors also intend to further modify the CI algorithm to solve complex NP-hard bilevel programming problems from supply chain optimization domain [32]. Also, it is quite important to tune up the learning rate of CI candidates so as to apply to dynamic control systems [33]. The ability of CI in clustering [34, 35] and classification domain in association with the cross-border transportation system and goods consolidation is currently underway.

Acknowledgments The authors would like to thank Abdesslem Layeb from MISC Lab, Computer Science Department, University of Constantine, Algeria for his useful discussion in the regard of the Knapsack Problem test cases.

Appendix

f_{11} N = 30, W = 577

w = {46, 17, 35, 1, 26, 17, 17, 48, 38, 17, 32, 21, 29, 48, 31, 8, 42, 37, 6, 9, 15, 22, 27, 14, 42, 40, 14, 31, 6, 34}

v = {57, 64, 50, 6, 52, 6, 85, 60, 70, 65, 63, 96, 18, 48, 85, 50, 77, 18, 70, 92, 17, 43, 5, 23, 67, 88, 35, 3, 91, 48}

f_{12} N = 35, W = 655

w = {7, 4, 36, 47, 6, 33, 8, 35, 32, 3, 40, 50, 22, 18, 3, 12, 30, 31, 13, 33, 4, 48, 5, 17, 33, 26, 27, 19, 39, 15, 33, 47, 17, 41, 40}

v = {35, 67, 30, 69, 40, 40, 21, 73, 82, 93, 52, 20, 61, 20, 42, 86, 43, 93, 38, 70, 59, 11, 42, 93, 6, 39, 25, 23, 36, 93, 51, 81, 36, 46, 96}

f_{13} N = 40, W = 819

w = {28, 23, 35, 38, 20, 29, 11, 48, 26, 14, 12, 48, 35, 36, 33, 39, 30, 26, 44, 20, 13, 15, 46, 36, 43, 19, 32, 2, 47, 24, 26, 39, 17, 32, 17, 16, 33, 22, 6, 12}

v = {13, 16, 42, 69, 66, 68, 1, 13, 77, 85, 75, 95, 92, 23, 51, 79, 53, 62, 56, 74, 7, 50, 23, 34, 56, 75, 42, 51, 13, 22, 30, 45, 25, 27, 90, 59, 94, 62, 26, 11}

f_{14} N = 45, W = 907

w = {18, 12, 38, 12, 23, 13, 18, 46, 1, 7, 20, 43, 11, 47, 49, 19, 50, 7, 39, 29, 32, 25, 12, 8, 32, 41, 34, 24, 48, 30, 12, 35, 17, 38, 50, 14, 47, 35, 5, 13, 47, 24, 45, 39, 1}

v = {98, 70, 66, 33, 2, 58, 4, 27, 20, 45, 77, 63, 32, 30, 8, 18, 73, 9, 92, 43, 8, 58, 84, 35, 78, 71, 60, 38, 40, 43, 43, 22, 50, 4, 57, 5, 88, 87, 34, 98, 96, 99, 16, 1, 25}

f_{15} N = 50, W = 882

w = {15, 40, 22, 28, 50, 35, 49, 5, 45, 3, 7, 32, 19,16, 40, 16, 31, 24, 15, 42, 29, 4, 14, 9, 29, 11, 25, 37, 48, 39, 5, 47, 49, 31, 48, 17, 46, 1, 25, 8, 16, 9, 30, 33, 18, 3, 3, 3, 4, 1}

v = {78, 69, 87, 59, 63, 12, 22, 4, 45, 33, 29, 50, 19, 94, 95, 60, 1, 91, 69, 8, 100, 32, 81, 47, 59, 48, 56, 18, 59, 16, 45, 54, 47, 84, 100, 98, 75, 20, 4, 19, 58, 63, 37, 64, 90, 26, 29, 13, 53, 83}

f_{16} N = 55, W = 1050

w = {27, 15, 46, 5, 40, 9, 36, 12, 11, 11, 49, 20, 32, 3, 12, 44, 24, 1, 24, 42, 44, 16, 12, 42, 22, 26, 10, 8, 46, 50, 20, 42, 48, 45, 43, 35, 9, 12, 22, 2, 14, 50, 16, 29, 31, 46, 20, 35, 11, 4, 32, 35, 15, 29, 16}

v = {98, 74, 76, 4, 12, 27, 90, 98, 100, 35, 30, 19, 75, 72, 19, 44, 5, 66, 79, 87, 79, 44, 35, 6, 82, 11, 1, 28, 95, 68, 39, 86, 68, 61, 44, 97, 83, 2, 15, 49, 59, 30, 44, 40, 14, 96, 37, 84, 5, 43, 8, 32, 95, 86, 18}

f_{17} N = 60, W = 1006

w = {7, 13, 47, 33, 38, 41, 3, 21, 37, 7, 32, 13, 42, 42, 23, 20, 49, 1, 20, 25, 31, 4, 8, 33, 11, 6, 3, 9, 26, 44, 39, 7, 4, 34, 25, 25, 16, 17, 46, 23, 38, 10, 5, 11, 28, 34, 47, 3, 9, 22, 17, 5, 41, 20, 33, 29, 1, 33, 16, 14}

v = {81, 37, 70, 64, 97, 21, 60, 9, 55, 85, 5, 33, 71, 87, 51, 100, 43, 27, 48, 17, 16, 27, 76, 61, 97, 78, 58, 46, 29, 76, 10, 11, 74, 36, 59, 30, 72, 37, 72, 100, 9, 47, 10, 73, 92, 9, 52, 56, 69, 30, 61, 20, 66, 70, 46, 16, 43, 60, 33, 84}

f_{18} N = 65, W = 1319

w = {47, 27, 24, 27, 17, 17, 50, 24, 38, 34, 40, 14, 15, 36, 10, 42, 9, 48, 37, 7, 43, 47, 29, 20, 23, 36, 14, 2, 48, 50, 39, 50, 25, 7, 24, 38, 34, 44, 38, 31, 14, 17, 42, 20, 5, 44, 22, 9, 1, 33, 19, 19, 23, 26, 16, 24, 1, 9, 16, 38, 30, 36, 41, 43, 6}

v = {47, 63, 81, 57, 3, 80, 28, 83, 69, 61, 39, 7, 100, 67, 23, 10, 25, 91, 22, 48, 91, 20, 45, 62, 60, 67, 27, 43, 80, 94, 47, 31, 44, 31, 28, 14, 17, 50, 9, 93, 15, 17, 72, 68, 36, 10, 1, 38, 79, 45, 10, 81, 66, 46, 54, 53, 63, 65, 20, 81, 20, 42, 24, 28, 1}

f_{19} N = 70, W = 1426

$$\mathrm{w} = \{4,16,16,2,9,44,33,43,14,45,11,49,21,12,41,19,26,38,42,20,5,14,40,47,29,47,30,50,39,10,26,33,44,31,50,7,15,24,7,12,10,34,17,40,28,12,35,3,29,50,19,28,47,13,42,9,44,14,43,41,10,49,13,39,41,25,46,6,7,43\}$$

$$\mathrm{v} = \{66,76,71,61,4,20,34,65,22,8,99,21,99,62,25,52,72,26,12,55,22,32,98,31,95,42,2,32,16,100,46,55,27,89,11,83,43,93,53,88,36,41,60,92,14,5,41,60,92,30,55,79,33,10,45,3,68,12,20,54,63,38,61,85,71,40,58,25,73,35\}$$

f_{20} N = 75, W = 1433

$$\mathrm{w} = \{24,45,15,40,9,37,13,5,43,35,48,50,27,46,24,45,2,7,38,20,20,31,2,20,3,35,27,4,21,22,33,11,5,24,37,31,46,13,12,12,41,36,44,36,34,22,29,50,48,17,8,21,28,2,44,45,25,11,37,35,24,9,40,45,8,47,1,22,1,12,36,35,14,17,5\}$$

$$\mathrm{v} = \{2,73,82,12,49,35,78,29,83,18,87,93,20,6,55,1,83,91,71,25,59,94,90,61,80,84,57,1,26,44,44,88,7,34,18,25,73,29,24,14,23,82,38,67,94,43,61,97,37,67,32,89,30,30,91,50,21,3,18,31,97,79,68,85,43,71,49,83,44,86,1,100,28,4,16\}$$

References

1. Deb K (2000) An efficient constraint handling method for genetic algorithms. Comput Meth Appl Mech Eng 186:311–338
2. Ray T, Tai K, Seow KC (2001) Multiobjective design optimization by an evolutionary algorithm. Eng Optim 33(4):399–424
3. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks, pp 1942–1948 (1995)
4. Dorigo M, Birattari M, Stitzle T (2006) Ant colony optimization: artificial ants as a computational intelligence technique. IEEE Comput Intell Mag 1:28–39
5. Pham DT, Ghanbarzadeh A, Koc E, Otri S, Rahim S, Zaidi M (2005) The Bees algorithm, technical note. Manufacturing Engineering Centre, Cardiff University
6. Feng Y, Jia K, He Y (2014) An improved hybrid encoding cuckoo search algorithm for 0–1 Knapsack Problems. Comput Intell Neurosci 2014:1–9
7. Kulkarni AJ, Durugkar IP, Kumar MR (2013) Cohort intelligence: a self supervised learning behavior. In: Proc. of IEEE international conference on systems, man, and cybernetics, pp 396–400
8. Liu L, Zhong WM, Qian F (2010) An improved chaos-particle swarm optimization algorithm. J East China Univ Sci Technol (Nat Sci Ed) 36:267–272
9. Xu W, Geng Z, Zhu Q, Gu X (2013) A piecewise linear chaotic map and sequential quadratic programming based robust hybrid particle swarm optimization. Inf Sci 218:85–102
10. Kennedy J (2003) Bare bones particle swarms. In: Proceedings of the IEEE swarm intelligence symposium (SIS'03), pp 80–87
11. Mendes R, Kennedy J, Neves J (2004) The fully informed particle swarm: simpler. Maybe better. IEEE Trans Evol Comput 8(3):204–210
12. Chen L, Sun H, Wang S (2009) Solving continuous optimization using ant colony algorithm. In: Second international conference on future information technology and management engineering, pp 92–95
13. Dorigo M, Maniezzo V, Colorni A (1996) Ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern Part B Cybern 26(1):29–41
14. Krishnasamy G, Kulkarni AJ, Paramesran R (2014) A hybrid approach for data clustering based on modified cohort intelligence and K-means. Exp Syst Appl 41(3):6009–6016
15. Hernández PR, Dimopoulos NJ (2005) A new heuristic for solving the multichoice multidimensional Knapsack Problem. In: Proc. of IEEE transactions on systems, man, and cybernetics—part A : systems and humans, vol 35, no 5, pp 708–717
16. Martello S, Toth P (1990) Knapsack Problems: algorithms and computer implementations. Wiley, New York, pp 1–296 (1990)
17. Tavares J, Pereira FB, Costa E (2008) Multidimensional Knapsack Problem: a fitness landscape analysis. In: Proc. of IEEE transactions on systems, man, and cybernetics—part B. Cybernetics 38(3):604–616
18. Moser M (1996) Declarative scheduling for optimally graceful QoS degradation. In: Proc. IEEE international conference multimedia computing systems, pp 86–93 (1996)
19. Khan MS (1998) Quality adaptation in a multisession multimedia system: model, algorithms and architecture. Ph.D. dissertation, Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada (1998)
20. Warner D, Prawda J (1972) A mathematical programming model for scheduling nursing personnel in a hospital. Manag Sci (Application Series Part 1) 19(4):411–422
21. Psinger D (1995) Algorithms for Knapsack Problems. PhD thesis, Department of Computer Science, University of Copenhagen, Copenhagen, Denmark (1995)
22. Laporte G (1992) The vehicle routing problem: an overview of exact and approximate algorithms. Eur J Oper Res 59:345–358
23. Granmo OC, Oommen BJ, Myrer SA, Olsen MG (2007) Learning automata-based solutions to the nonlinear fractional Knapsack Problem with applications to optimal resource allocation. Proc IEEE Trans Syst Man Cybern Part B Cybern 37(1):166–175
24. Zou D, Gao L, Li S, Wu J (2011) Solving 0–1 Knapsack Problem by a Novel Global Harmony Search algorithm. Appl Soft Comput 11:1556–1564
25. Layeb A (2013) A hybrid Quantum Inspired Harmony Search algorithm for 0–1 optimization problems. J Comput Appl Math 253:14–25
26. Layeb A (2011) A novel Quantum Inspired Cuckoo Search for Knapsack Problems. Int J Bio-Inspired Comput 3(5):297–305
27. Geem ZW, Kim JH, Loganathan GV (2001) A new heuristic optimization algorithm: harmony search. Simulation 76(2):60–68
28. Mahdavi M, Fesanghary M, Damangir E (2007) An Improved Harmony Search algorithm for solving optimization problems. Appl Math Comput 188:1567–1579
29. Deb K (2000) An efficient constraint handling method for genetic algorithms. Comput Meth Appl Mech Eng 186:311–338
30. Kulkarni AJ, Tai K (2011) Solving constrained optimization problems using probability collectives and a penalty function approach. Int J Comput Intell Appl 10(4):445–470
31. Kulkarni AJ, Tai K (2011) A probability collectives approach with a feasibility-based rule for constrained optimization. Appl Comput Intell Soft Comput 2011, Article ID 980216

32. Ma W, Wang M, Zhu X (2014) Improved particle swarm optimization based approach for bilevel programming problem-an application on supply chain model. Int J Mach Learn Cybernet 5(2):281–292
33. Chen CJ (2012) Structural vibration suppression by using neural classifier with genetic algorithm. Int J Mach Learn Cybernet 3(3):215–221
34. Wang XZ, He Q, Chen DG, Yeung D (2005) A genetic algorithm for solving the inverse problem of support vector machines. Neurocomputing 68:225–238
35. Wang XZ, He YL, Dong LC, Zhao HY (2011) Particle swarm optimization for determining fuzzy measures from data. Inf Sci 181(19):4230–4252