Chapter 5
Solution to 0–1 Knapsack Problem Using Cohort Intelligence Algorithm

The previous chapters discussed the algorithm Cohort Intelligence (CI) and its applicability solving several unconstrained and constrained problems. In addition CI was also applied for solving several clustering problems. This validated the learning and self supervising behavior of the cohort. This chapter further tests the ability of CI 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.

5.1 Knapsack Problem Using CI Method

The Knapsack Problem (KP) 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 [1, 2]. The special case of single constraints is generally known as the KP or the Uni-dimensional KP [3]. 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 [4]. Khan [5] used the MMKP 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 [6], 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 [7], 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. [8]. Apart from these applications, KPs are being used for resource allocation problems dealing with the World Wide Web [9]. In this chapter various cases of the 0–1 KP [10–12] were solved using CI. In all the problems, the implemented CI methodology produced robust results with reasonable computational cost.

The problem is described as follows [10–14]: given a set of N objects, each object i, $i = 1, \ldots, N$ 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

$$f(\mathbf{w}) = \sum_{i=1}^{N} w_i x_i, x_i \in \{0, 1\}, \quad 1 \le i \le N \tag{5.1}$$

5.1.1 Illustration of CI Solving 0–1 KP

In the context of CI algorithm (discussed in Sect. 5.1), the objects $i, i = 1, \ldots, N$ 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 $\mathbf{F}^C = \{f(\mathbf{v}^1), \ldots, f(\mathbf{v}^c), \ldots, f(\mathbf{v}^C)\}$ and weights $\mathbf{F}^{CW} = \{f(\mathbf{w}^1), \ldots, f(\mathbf{w}^c), \ldots, f(\mathbf{w}^C)\}$ 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

$$p_1^c = \frac{f(\mathbf{v}^c)}{\sum_{c=1}^{C} f(\mathbf{v}^c)} \tag{5.2}$$

and

$$p_2^c = \begin{cases} \frac{f(\mathbf{w}^c)}{W} & f(\mathbf{w}^c) \leq W \\ 3 - \frac{2f(\mathbf{w}^c)}{W} & f(\mathbf{w}^c) > W \end{cases} \tag{5.3}$$

A probability distribution specially devised to bias the solution towards feasibility is represented in Fig. 5.1. 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 $f\left(\mathbf{v}^{c[?]}\right)$ 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 [15–17]. The modifications are categorized as follows:

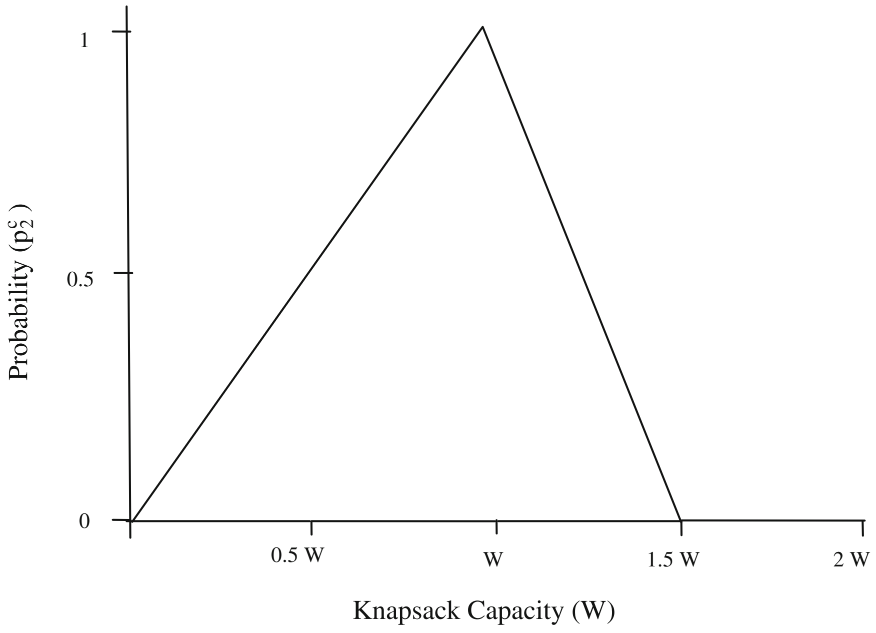


Fig. 5.1 Probability distribution for p_2^c

1. If the solution of candidate c $(c = 1, \ldots, C)$ is feasible i.e. it satisfies the weight constraint given by Eq. 5.1 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. 5.1 is still satisfied.
   1.2. Replaces a randomly chosen object with another randomly chosen one from the candidate being followed, such that Eq. 5.1 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 $\mathbf{F}^{c,t} = \left\{f(\mathbf{v}^c)^1, \ldots, f(\mathbf{v}^c)^j, \ldots, f(\mathbf{v}^c)^t\right\}, \quad (c = 1, \ldots, C)$. 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 $\mathbf{F}^C = \{f^*(\mathbf{v}^1), \ldots, f^*(\mathbf{v}^c), \ldots, f^*(\mathbf{v}^C)\}$.

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.

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 $w_i, \quad i = 1, \ldots, N$ and profits $v_i, \quad i = 1, \ldots, N$ associated with every object are illustrated in Fig. 5.2. 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 $\mathbf{F}^C = \left\{f(\mathbf{v}^1), \ldots, f(\mathbf{v}^c), \ldots, f(\mathbf{v}^C)\right\}$ and weights

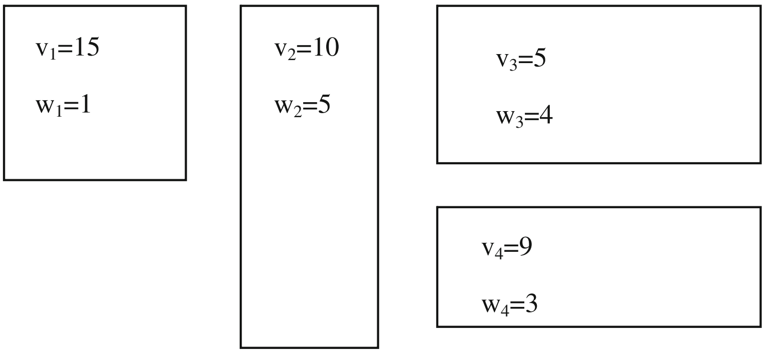


Fig. 5.2 Illustrative 0–1 KP example with $N = 4, W = 8$

$\mathbf{F}^{C,W} = \{f(\mathbf{w}^1), \ldots, f(\mathbf{w}^c), \ldots, f(\mathbf{w}^C)\}$ are calculated. The further CI algorithm steps are discussed below:

(1) The probability p^c associated with each candidate $c(c = 1, \ldots, 3)$ is calculated using Eqs. 5.2 and 5.3. The calculated probability values are presented in Fig. 5.3.
(2) Using roulette wheel selection approach, assume that candidate 1 decides to follow candidate 3. As presented in Fig. 5.4, $t = 3$ variations are formed along

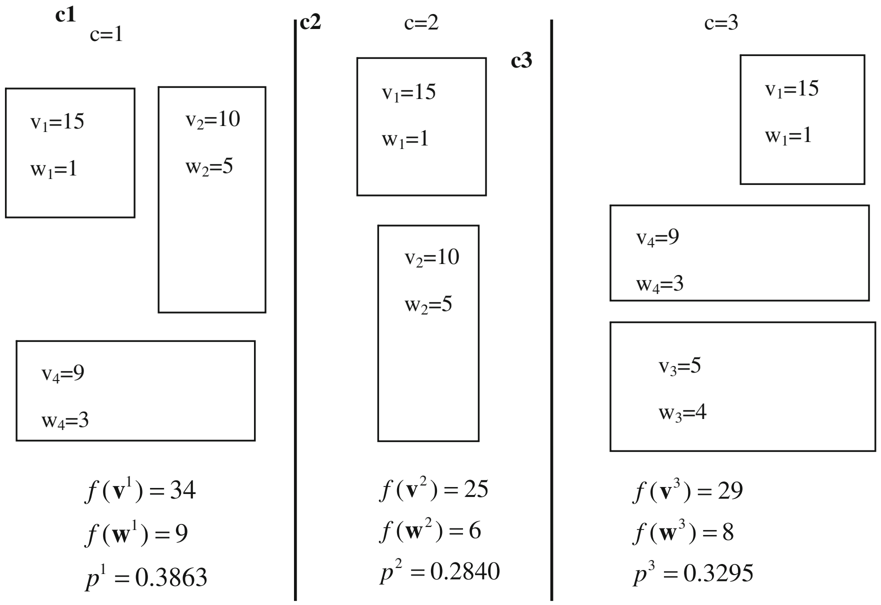


Fig. 5.3 Illustrative 0–1 KP example with $C = 3$

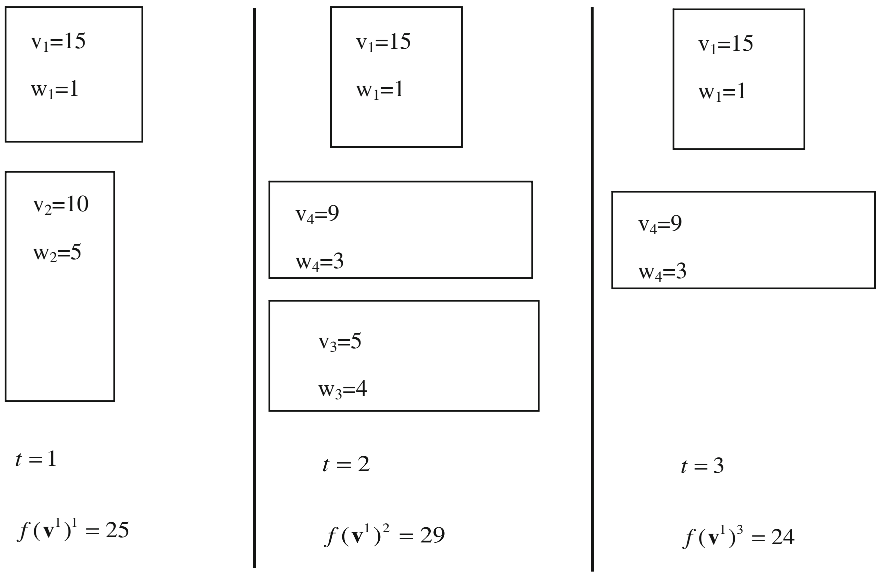
with the associated profits vector $\mathbf{F}^{1,3} = \left\{f(\mathbf{v}^1)^1, f(\mathbf{v}^1)^2, f(\mathbf{v}^1)^3\right\}$ and the selected variation with profit $f^*(\mathbf{v}^1)$. In this way, candidates 2 and 3 also follow certain candidate and update themselves. It makes the cohort available with 3 updated candidates with profits $\mathbf{F}^3 = \left\{f^*(\mathbf{v}^1), f^*(\mathbf{v}^2), f^*(\mathbf{v}^3)\right\}$. This

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 $f\left(\mathbf{v}^1\right)^1$: 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



$$\mathbf{F}^{1,3} = \left\{f\left(\mathbf{v}^1\right)^1, f\left(\mathbf{v}^1\right)^2, f\left(\mathbf{v}^1\right)^3\right\} = \{25, 29, 24\}$$

$$f^*(\mathbf{v}^1) = 29$$

Fig. 5.4 Illustrative 0–1 KP example with $t = 3$ (variations obtained)

process continues until saturation (convergence), i.e. every candidate finds the solution and does not change for successive considerable number of learning attempts.

5.2 Results and Discussion

The CI algorithm discussed in Sect. 5.1 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 $f_1 - f_{10}$ [10–12] are presented in Table 5.1. The cases $f_{11} - f_{20}$ were generated using a random number generator. In these tests, knapsack capacity is calculated using the formula [11, 12]: $W = \frac{3}{4}\sum_{i=1}^{N} w_i$ 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 at the end of this chapter.

Recently, the instances $f_1 - f_{10}$ were solved using Harmony Search (HS) [10, 13], Improved Harmony Search (IHS) [10, 14], Novel Global Harmony Search (NGHS) [10–12], Quantum Inspired Cuckoo Search Algorithm QICSA [12], and Quantum Inspired Harmony Search Algorithm (QIHSA) [11]. 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 [10]. 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. Zou et al. [10] used a penalty function method 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 [12]. 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 [11]. 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.

Table 5.1 The dimension and parameters of ten test problems

f	Number of objects (N)	Parameters $(W, \mathbf{w}, \mathbf{v})$
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	$W = 878,\ \mathbf{w} = \{92, 4, 43, 83, 84, 68, 92, 82, 6, 44, 32, 18, 56, 83, 25, 96, 70, 48, 14, 58\},$ $\mathbf{v} = \{44, 46, 90, 72, 91, 40, 75, 35, 8, 54, 78, 40, 7, 15, 61, 17, 75, 29, 75, 63\}$
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	$W = 10{,}000,\ \mathbf{w} = \{983, 982, 981, 980, 979, 978, 488, 976, 972, 486, 486, 972, 972, 485, 485, 969, 966, 483, 964, 963, 961, 958, 959\},$ $\mathbf{v} = \{981, 980, 979, 978, 977, 976, 487, 974, 970, 485, 485, 970, 970, 484, 484, 976, 974, 482, 962, 961, 959, 958, 857\}$

(continued)

Table 5.1 (continued)

f	Number of objects (N)	Parameters $(W, \mathbf{w}, \mathbf{v})$
f_9	5	$W = 80,\ \mathbf{w} = \{15, 20, 17, 8, 31\}$, $\mathbf{v} = \{33, 24, 36, 37, 12\}$
f_{10}	20	$W = 879,\ \mathbf{w} = \{84, 83, 43, 4, 44, 6, 82, 92, 25, 83, 56, 18, 58, 14, 48, 70, 96, 32, 68, 92\}$, $\mathbf{v} = \{91, 72, 90, 46, 55, 8, 35, 75, 61, 15, 77, 40, 63, 75, 29, 75, 17, 78, 40, 44\}$

The approach of CI handles constraints using a probability distribution p_2^c (refer to Fig. 5.1) 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 [15–17] was also incorporated which assists candidates select the variations with better objective and

Table 5.2 Summary of solutions of KPs solved using CI

Problem	Number of objects, Knapsack capacity (N, W)	Solution $(f^*(\mathbf{v}), f^*(\mathbf{w}))$			Standard deviation	Average function evaluations (FE)	Average time (s)	Parameters (C, t)
		Best	Mean	Worst				
f_1	10, 269	295, 269	267.46, 262.722	260, 250	0.0	5410	0.4489	5, 10
f_2	20, 878	1024, 871	1020.55, 852.84	1009, 827	0.0	5446	1.5909	5, 10
f_3	4, 20	35, 18	34.55, 17.867	28, 16	0.0	5136	0.2687	5, 10
f_4	4, 11	23, 11	22.06, 10.33	16, 6	0.64	5193	0.2492	5, 10
f_5	15, 375	481.0694, 354.9608	449.986, 361.692	412.6988, 372.9118	10.68	5590	0.6609	5, 10
f_6	10, 60	51, 56	50.733, 56.733	49, 54	0.66	5573	0.4465	5, 10
f_7	7, 50	105, 50	86.6, 44.8	79, 42	2.99	5696	0.3749	5, 10
f_8	23, 10,000	9759, 9760	9753.33, 9756.33	9710, 9711	11.5	6486	1.1959	5, 10
f_9	5, 80	130, 60	124.6, 61.4	106, 74	2.89	5110	0.3048	5, 10
f_{10}	20, 879	1025, 871	997.7, 558.3	892, 805	18.6	5426	1.535	5, 10
f_{11}	30, 577	1437, 566	1418, 571.5	1398, 563	11.79	6817	3.4635	5, 10
f_{12}	35, 655	1689, 650	1686.5, 650.833	1679, 654	3.8188	5375	5.2288	5, 10
f_{13}	40, 819	1816, 817	1807.5, 817.66	1791, 819	9.604	7833	7.3429	5, 10
f_{14}	45, 907	2020, 903	2017, 902.5	2007, 901	4.749	7433	8.1510	5, 10
f_{15}	50, 882	2440, 873	2436.166, 870.33	2421, 865	6.841	7766	10.5690	5, 10
f_{16}	55, 1050	2643, 1049	2605, 1047.8	2581, 1049	22.018	9720	14.3445	5, 10
f_{17}	60, 1006	2917, 1002	2915, 1001.833	2905, 1001	4.472	9017	17.0894	5, 10
f_{18}	65, 1319	2814, 1319	2773.66, 1316.33	2716, 1317	18.273	10,283	20.9486	5, 10
f_{19}	70, 1426	3221, 1426	3216, 1423.166	3211, 1419	4.3589	10,333	26.4846	5, 10
f_{20}	75, 1433	3614, 1432	3603.8, 1431.8	3591, 1429	8.035	12,720	34.0072	5, 10

Table 5.3 Comparison of results obtained using CI with other established methods

Problem	Number of objects (N)	Method	Optimum solution $f^*(\mathbf{v})$
f_1	10	HS [10]	295
		IHS [10]	295
		NGHS [10, 11]	295
		QICSA [11]	295
		QIHSA [12]	295
		CI	295
f_2	20	HS [10]	1024
		IHS [10]	1024
		NGHS [10, 11]	1024
		QICSA [11]	1024
		QIHSA [12]	1024
		CI	1024
f_3	4	HS [10]	35
		IHS [10]	35
		NGHS [10, 11]	35
		QICSA [11]	35
		QIHSA [12]	35
		CI	35
f_4	4	HS [10]	23
		IHS [10]	23
		NGHS [10, 11]	23
		QICSA [11]	23
		QIHSA [12]	23
		CI	23
f_5	15	HS [10]	481.0694
		IHS [10]	481.0694
		NGHS [10, 11]	481.0694
		QICSA [11]	481.0694
		QIHSA [12]	481.0694
		CI	481.0694
f_6	10	HS [10]	50
		IHS [10]	50
		NGHS [10, 11]	52
		QICSA [11]	52
		QIHSA [12]	52
		CI	51
f_7	7	HS [10]	107
		IHS [10]	107
		NGHS [10, 11]	107
		QICSA [11]	107
		QIHSA [12]	107
		CI	105
f_8	23	HS [10]	9767
		IHS [10]	9767
		NGHS [10, 11]	9767
		QICSA [11]	9767
		QIHSA [12]	9767
		CI	9759

(continued)

Table 5.3 (continued)

Problem	Number of objects (N)	Method	Optimum solution $f^*(\mathbf{v})$
f_9	5	HS [10]	130
		IHS [10]	130
		NGHS [10, 11]	130
		QICSA [11]	130
		QIHSA [12]	130
		CI	130
f_{10}	20	HS [10]	1025
		IHS [10]	1025
		NGHS [10, 11]	1025
		QICSA [11]	1025
		QIHSA [12]	1025
		CI	1025

constraint satisfaction (refer to Sect. 5.1). 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 5.2. In addition, the CI parameters such as number of candidates C and number of variations t are also listed. As presented in Table 5.3, 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 5.2, 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 5.3, 5.4 and Fig. 5.5 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. 5.6 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 by Tavares et al. [17], 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 discussed by Tavares et al. [17] where several parameters were required to be tuned based on the preliminary trials.

Table 5.4 Comparison of results obtained using CI with B&B

Problem	Number of objects (N)	Optimum solution $(f^*(\mathbf{v}))$ CI B&B	Time (s)
f_1	10	295 295	0.4489 0.12
f_2	20	1024 1024	1.5909 0.04
f_3	4	35 35	0.2687 0.03
f_4	4	23 23	0.2492 0.03
f_5	15	481.0694 481.0690	0.6609 0.18
f_6	10	51 52	0.4465 0.14
f_7	7	105 107	0.3749 0.04
f_8	23	9759 9767	1.1959 0.18
f_9	5	130 130	0.3048 0.03
f_{10}	20	1025 1025	1.535 0.45
f_{11}	30	1437 1437	3.4635 0.156001
f_{12}	35	1689 1689	5.2288 0.0624004
f_{13}	40	1816 1821	7.3429 0.0156001
f_{14}	45	2020 2033	8.1510 0.0312002
f_{15}	50	2440 2440	10.5690 0.0312002
f_{16}	55	2643 2440	14.3445 0.0312002
f_{17}	60	2917 2917	17.0894 0.0312002
f_{18}	65	2814 2818	20.9486 0.0624004
f_{19}	70	3221 3223	26.4846 0.0780005
f_{20}	75	3614 3614	34.0072 0.0312002

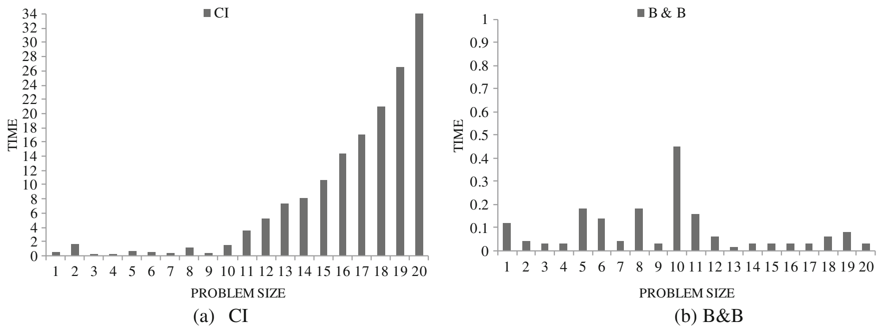


Fig. 5.5 Effect of problem size (N) on computational time

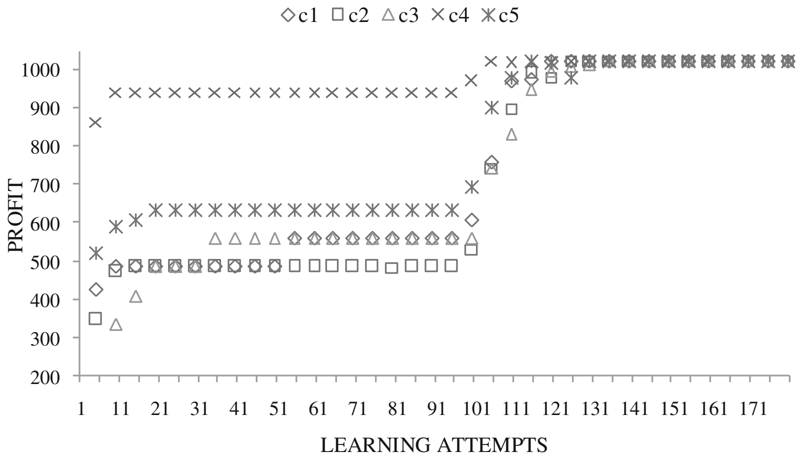


Fig. 5.6 Learning attempts versus the objective function $f(v)^*$ for each candidate

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. 5.7, 5.8 and 5.9, respectively. Another important observation was that as the problem size i.e.

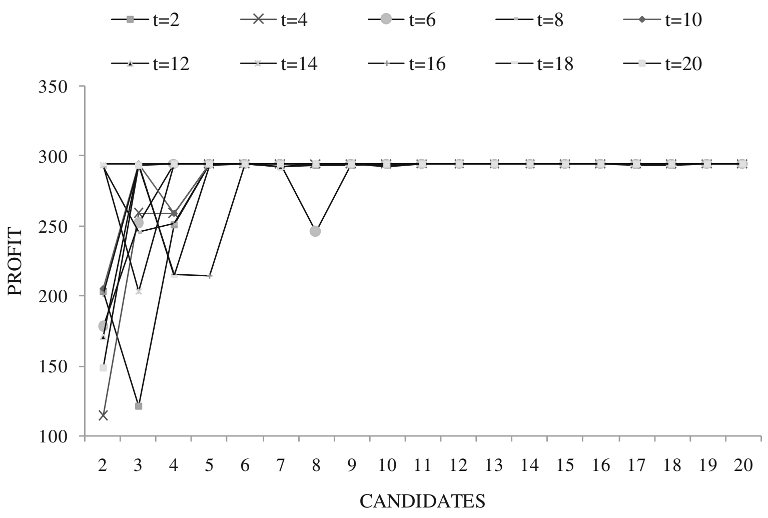


Fig. 5.7 Effect of number of candidates (C) on the profit $f(\mathbf{v})^*$ for different values of variations (t)

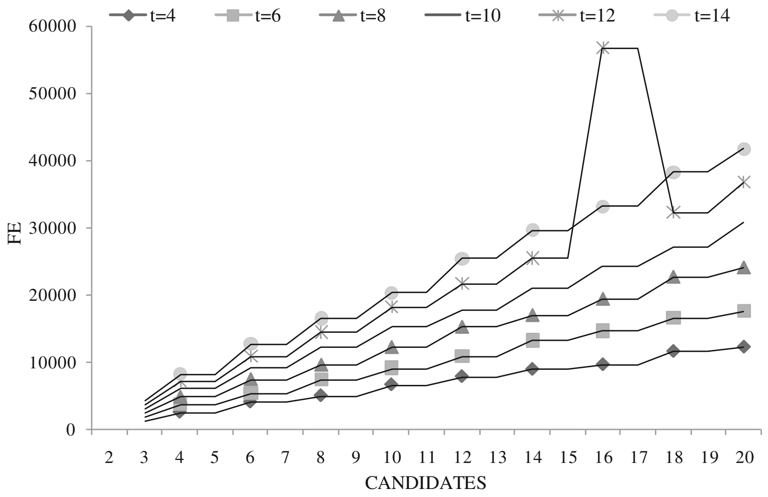


Fig. 5.8 Effect of number of candidates (C) on the function evaluations (FE) for different values of variations (t)

N increased, the computational cost also increased (refer to Table 5.2). 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

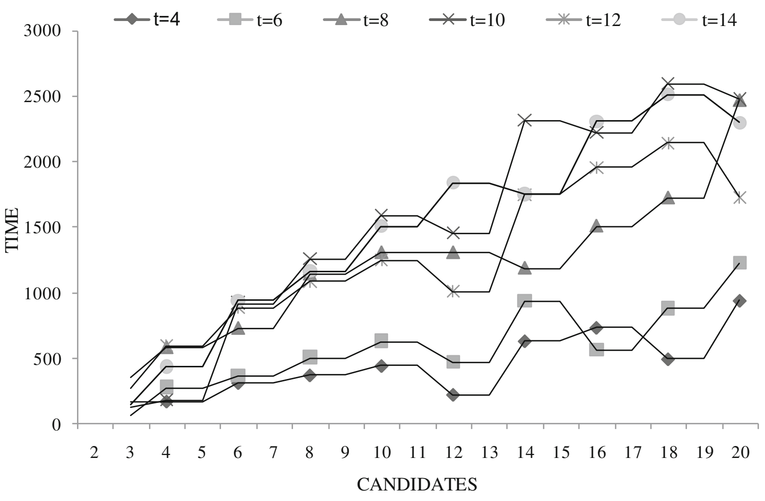


Fig. 5.9 Effect of number of candidates (C) on the time for different values of variations (t)

which there was no significant change (refer to Fig. 5.7). 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.

5.3 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 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.

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 [18]. Also, it is quite important to tune up the learning rate of CI candidates so as to apply to dynamic control systems [19]. The ability of CI in clustering [20–22] and classification domain in association with the cross-border transportation system and goods consolidation is currently underway.

5.4 Test Cases

f_{11}. N = 30, W = 577

$$\begin{aligned}\mathrm{w} &= \{46, 17, 35, 1, 26, 17, 17, 48, 38, 17, 32, 21, 29, 48, 31,\\ &\quad 8, 42, 37, 6, 9, 15, 22, 27, 14, 42, 40, 14, 31, 6, 34\}\\ \mathrm{v} &= \{57, 64, 50, 6, 52, 6, 85, 60, 70, 65, 63, 96, 18, 48, 85,\\ &\quad 50, 77, 18, 70, 92, 17, 43, 5, 23, 67, 88, 35, 3, 91, 48\}\end{aligned}$$

f_{12}. N = 35, W = 655

$$\begin{aligned}\mathrm{w} &= \{7, 4, 36, 47, 6, 33, 8, 35, 32, 3, 40, 50, 22, 18, 3, 12, 30, 31,\\ &\quad 13, 33, 4, 48, 5, 17, 33, 26, 27, 19, 39, 15, 33, 47, 17, 41, 40\}\\ \mathrm{v} &= \{35, 67, 30, 69, 40, 40, 21, 73, 82, 93, 52, 20, 61, 20, 42, 86, 43,\\ &\quad 93, 38, 70, 59, 11, 42, 93, 6, 39, 25, 23, 36, 93, 51, 81, 36, 46, 96\}\end{aligned}$$

f_{13}. N = 40, W = 819

$$\begin{aligned} \mathrm{w} = \{&28, 23, 35, 38, 20, 29, 11, 48, 26, 14, 12, 48, 35, 36, 33, 39, 30, 26,\\ &44, 20, 13, 15, 46, 36, 43, 19, 32, 2, 47, 24, 26, 39, 17, 32, 17, 16, 33, 22, 6, 12\}\\ \mathrm{v} = \{&13, 16, 42, 69, 66, 68, 1, 13, 77, 85, 75, 95, 92, 23, 51, 79, 53, 62, 56, 74,\\ &7, 50, 23, 34, 56, 75, 42, 51, 13, 22, 30, 45, 25, 27, 90, 59, 94, 62, 26, 11\} \end{aligned}$$

f_{14}. N = 45, W = 907

$$\begin{aligned} \mathrm{w} = \{&18, 12, 38, 12, 23, 13, 18, 46, 1, 7, 20, 43, 11, 47, 49, 19, 50, 7, 39, 29, 32, 25, 12,\\ &8, 32, 41, 34, 24, 48, 30, 12, 35, 17, 38, 50, 14, 47, 35, 5, 13, 47, 24, 45, 39, 1\}\\ \mathrm{v} = \{&98, 70, 66, 33, 2, 58, 4, 27, 20, 45, 77, 63, 32, 30, 8, 18, 73, 9, 92, 43, 8, 58, 84,\\ &35, 78, 71, 60, 38, 40, 43, 43, 22, 50, 4, 57, 5, 88, 87, 34, 98, 96, 99, 16, 1, 25\} \end{aligned}$$

f_{15}. N = 50, W = 882

$$\begin{aligned} \mathrm{w} = \{&15, 40, 22, 28, 50, 35, 49, 5, 45, 3, 7, 32, 19, 16, 40, 16, 31, 24, 15, 42,\\ &29, 4, 14, 9, 29, 11, 25, 37, 48, 39, 5, 47, 49, 31, 48, 17,\\ &46, 1, 25, 8, 16, 9, 30, 33, 18, 3, 3, 3, 4, 1\}\\ \mathrm{v} = \{&78, 69, 87, 59, 63, 12, 22, 4, 45, 33, 29, 50, 19, 94, 95, 60, 1, 91, 69, 8,\\ &100, , 84, 100, 32, 81, 47, 59, 48, 56, 18, 59, 16, 45, 54, 4798, 75, 20,\\ &4, 19, 58, 63, 37, 64, 90, 26, 29, 13, 53, 83\} \end{aligned}$$

f_{16}. N = 55, W = 1050

$$\begin{aligned} \mathrm{w} = \{&27, 15, 46, 5, 40, 9, 36, 12, 11, 11, 49, 20, 32, 3, 12, 44, 24, 1, 24, 42,\\ &44, 16, 12, 42, 22, 26, 10, 8, 46, 50, 20, 42, 48, 45, 43, 35, 9, 12,\\ &22, 2, 14, 50, 16, 29, 31, 46, 20, 35, 11, 4, 32, 35, 15, 29, 16\}\\ \mathrm{v} = \{&98, 74, 76, 4, 12, 27, 90, 98, 100, 35, 30, 19, 75, 72, 19, 44, 5, 66,\\ &79, 87, 79, 44, 35, 6, 82, 11, 1, 28, 95, 68, 39, 86, 68, 61, 44, 97, 83, 2, 15,\\ &49, 59, 30, 44, 40, 14, 96, 37, 84, 5, 43, 8, 32, 95, 86, 18\} \end{aligned}$$

f_{17}. N = 60, W = 1006

$$\begin{aligned} w = \{&7, 13, 47, 33, 38, 41, 3, 21, 37, 7, 32, 13, 42, 42, 23, 20, 49, 1, 20, 25, 31, 4, 8, \\ &33, 11, 6, 3, 9, 26, 44, 39, 7, 4, 34, 25, 25, 16, 17, 46, 23, 38, 10, 5, 11, \\ &28, 34, 47, 3, 9, 22, 17, 5, 41, 20, 33, 29, 1, 33, 16, 14\} \\ v = \{&81, 37, 70, 64, 97, 21, 60, 9, 55, 85, 5, 33, 71, 87, 51, 100, 43, 27, 48, 17, 16, \\ &27, 76, 61, 97, 78, 58, 46, 29, 76, 10, 11, 74, 36, 59, 30, 72, 37, 72, 100, 9, 47, \\ &10, 73, 92, 9, 52, 56, 69, 30, 61, 20, 66, 70, 46, 16, 43, 60, 33, 84\} \end{aligned}$$

f_{18}. N = 65, W = 1319

$$\begin{aligned} w = \{&47, 27, 24, 27, 17, 17, 50, 24, 38, 34, 40, 14, 15, 36, 10, 42, 9, 48, 37, 7, 43, 47, 29, \\ &20, 23, 36, 14, 2, 48, 50, 39, 50, 25, 7, 24, 38, 34, 44, 38, 31, 14, 17, 42, 20, \\ &5, 44, 22, 9, 1, 33, 19, 19, 23, 26, 16, 24, 1, 9, 16, 38, 30, 36, 41, 43, 6\} \\ v = \{&47, 63, 81, 57, 3, 80, 28, 83, 69, 61, 39, 7, 100, 67, 23, 10, 25, 91, 22, 48, 91, 20, \\ &45, 62, 60, 67, 27, 43, 80, 94, 47, 31, 44, 31, 28, 14, 17, 50, 9, 93, 15, 17, 72, 68, 36, \\ &10, 1, 38, 79, 45, 10, 81, 66, 46, 54, 53, 63, 65, 20, 81, 20, 42, 24, 28, 1\} \end{aligned}$$

f_{19}. N = 70, W = 1426

$$\begin{aligned} w = \{&4, 16, 16, 2, 9, 44, 33, 43, 14, 45, 11, 49, 21, 12, 41, 19, 26, 38, 42, 20, \\ &5, 14, 40, 47, 29, 47, 30, 50, 39, 10, 26, 33, 44, 31, 50, 7, 15, 24, 7, 12, \\ &10, 34, 17, 40, 28, 12, 35, 3, 29, 50, 19, 28, 47, 13, 42, 9, 44, 14, 43, 41, \\ &10, 49, 13, 39, 41, 25, 46, 6, 7, 43\} \\ v = \{&66, 76, 71, 61, 4, 20, 34, 65, 22, 8, 99, 21, 99, 62, 25, 52, 72, 26, 12, 55, \\ &22, 32, 98, 31, 95, 42, 2, 32, 16, 100, 46, 55, 27, 89, 11, 8, 3, 43, 93, 53, 88, \\ &36, 41, 60, 92, 14, 5, 41, 60, 92, 30, 55, 79, 33, 10, 45, 3, 68, 12, 20, 54, 63, \\ &38, 61, 85, 71, 40, 58, 25, 73, 35\} \end{aligned}$$

f_{20}. N = 75, W = 1433

$$\begin{aligned} w = \{&24, 45, 15, 40, 9, 37, 13, 5, 43, 35, 48, 50, 27, 46, 24, 45, 2, 7, 38, 20, \\ &20, 31, 2, 20, 3, 35, 27, 4, 21, 22, 33, 11, 5, 24, 37, 31, 46, 13, 12, 12, \\ &41, 36, 44, 36, 34, 22, 29, 50, 48, 17, 8, 21, 28, 2, 44, 45, 25, 11, 37, 35, \\ &24, 9, 40, 45, 8, 47, 1, 22, 1, 12, 36, 35, 14, 17, 5\} \\ v = \{&2, 73, 82, 12, 49, 35, 78, 29, 83, 18, 87, 93, 20, 6, 55, 1, 83, 91, 71, 25, 59, \\ &94, 90, 61, 80, 84, 57, 1, 26, 44, 44, 88, 7, 34, 18, 25, 73, 29, 24, 14, 23, 82, \\ &38, 67, 94, 43, 61, 97, 37, 67, 32, 89, 30, 30, 91, 50, 21, 3, 18, 31, 97, 79, 68, \\ &85, 43, 71, 49, 83, 44, 86, 1, 100, 28, 4, 16\} \end{aligned}$$

References

1. Hernández, P.R., Dimopoulos, N.J.: A new heuristic for solving the multichoice multidimensional knapsack problem. Proc. IEEE Trans. Syst. Man Cybernet. Part A: Syst. Hum. **35**(5), 708–717 (2005)
2. Martello, S., Toth, P.: Knapsack Problems: Algorithms and Computer Implementations, pp. 1–296. Wiley, New York (1990)
3. Tavares, J., Pereira, F.B., Costa, E.: Multidimensional knapsack problem: a fitness landscape analysis. Proc. IEEE Trans. Syst. Man Cybernet. Part B: Syst. Hum. **38**(3), 604–616 (2008)
4. Moser, M.: Declarative scheduling for optimally graceful QoS degradation. In: Proceedings of IEEE International Conference Multimedia Computing Systems, pp. 86–93 (1996)
5. Khan, M.S.: 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)
6. Warner, D., Prawda, J.: A mathematical programming model for scheduling nursing personnel in a hospital. Manag. Sci. (Appl. Ser. Part 1) **19**(4), 411–422 (1972)
7. Psinger D.: Algorithms for Knapsack Problems, PhD thesis, Department of Computer Science, University of Copenhagen, Copenhagen,Denmark, (1995)
8. Laporte, G.: The vehicle routing problem: an overview of exact and approximate algorithms. Eur. J. Oper. Res. **59**, 345–358 (1992)
9. Granmo, O.C., Oommen, B.J., Myrer, S.A., Olsen, M.G.: Learning automata-based solutions to the nonlinear fractional knapsack problem with applications to optimal resource allocation. Proc. IEEE Trans. Syst. Man Cybernet. Part B Cybernet. **37**(1), 166–175 (2007)
10. Zou, D., Gao, L., Li, S., Wu, J.: Solving 0–1 knapsack problem by a novel global harmony search algorithm. Appl. Soft Comput. **11**, 1556–1564 (2011)
11. Layeb, A.: A hybrid quantum inspired harmony search algorithm for 0–1 optimization problems. J. Comput. Appl. Math. **253**, 14–25 (2013)
12. Layeb, A.: A novel quantum inspired cuckoo search for knapsack problems. Int. J. Bio-Inspir. Comput. **3**(5), 297–305 (2011)
13. Geem, Z.W., Kim, J.H., Loganathan, G.V.: A new heuristic optimization algorithm: harmony search. Simulation **76**(2), 60–68 (2001)
14. Mahdavi, M., Fesanghary, M., Damangir, E.: An improved harmony search algorithm for solving optimization problems. Appl. Math. Comput. **188**, 1567–1579 (2007)
15. Deb, K.: An efficient constraint handling method for genetic algorithms. Comput. Methods Appl. Mech. Eng. **186**, 311–338 (2000)
16. Kulkarni, A.J., Tai, K.: Solving constrained optimization problems using probability collectives and a penalty function approach. Int. J. Comput. Intell. Appl. **10**(4), 445–470 (2011)
17. Kulkarni, A.J., Tai, K.: A probability collectives approach with a feasibility-based rule for constrained optimization. Appl. Comput. Intell. Soft Comput. **2011**, Article ID 980216 (2011)
18. Ma, W., Wang, M.: Zhu X: 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 (2014)
19. Chen, C.J.: Structural vibration suppression by using neural classifier with genetic algorithm. Int. J. Mach. Learn. Cybernet. **3**(3), 215–221 (2012)
20. Wang, X.Z., He, Q., Chen, D.G., Yeung, D.: A genetic algorithm for solving the inverse problem of support vector machines. Neurocomputing **68**, 225–238 (2005)
21. Wang, X.Z., He, Y.L., Dong, L.C., Zhao, H.Y.: Particle swarm optimization for determining fuzzy measures from data. Inf. Sci. **181**(19), 4230–4252 (2011)
22. Krishnasamy, G., Kulkarni, A.J., Paramesran, R.: A hybrid approach for data clustering based on modified cohort intelligence and k-means. Expert Syst. Appl. **41**, 6009–6016 (2014)