# Chapter 7
# Solution to a New Variant of the Assignment Problem Using Cohort Intelligence Algorithm

**Nomenclature**

| | |
|---|---|
| $C$ | An $n$ by $n$ row circular matrix. The $ij$th element of $C$ is $C_{i,j}$, where $i,j = 1,\ldots,n$ |
| $N$ | The set of integers $\{1, 2, \ldots, n\}$ |
| $\pi$ | A permutation of set $N$ |
| $C^{\pi}$ | An $n$ by $n$ matrix obtained by shifting each element of row $i$ of matrix $C$ for $i = 1,\ldots,n$ by $(\pi(i) - 1)$ positions to the right in a circular manner. In other words, The $ik$th element of $C^{\pi}$ is given by $C^{\pi}_{i,k} = C_{i,k-\pi(i)+1} \forall\ 1 \le i \le n$, $1 \le k \le n$ |
| $I_k$ | The sum of the $k$th column of matrix $C^{\pi}$ |
| $Z$ | The maximum column sum of matrix $C^{\pi}$, $Z = max^{n}_{k=1}\{I_k\}$ |
| $x_{ij}$ | A binary variable equal to 1 if $\pi(i) = j$; and 0, otherwise |

In this chapter, we present a variant of the classical assignment problem [1]. The model has applications in healthcare systems and inventory management. The problem stems from an application in healthcare management. Specifically, a surgical scheduling in a hospital setting is a complex combinatorial problem. In addition, similar problem arises in minimizing the space requirements in a retail store. The problem formulation, applications and solution using Cohort Intelligence methodology [2–4] is presented in sections below.

## 7.1 New Variant of the Assignment Problem

Suppose we seek to schedule $n$ surgeons/doctors over a planning horizon of $n$ days. The recovery time for each operated patient in the recovery room varies according to the type of surgery. When building cyclic surgery schedules, one important objective is to minimize congestion in the recovery room. That is, we want to minimize the maximum number of patients in the recovery unit in any given day of the planning horizon so that the costs associated with important resources such as nurses, space, beds, and equipment are also minimized. Another application of this problem arises in supply chain management. By considering cyclic scheduling for suppliers, the maximum required storage space of a retail shop on any day over a planning horizon of $n$ days can be minimized by developing optimal delivery schedules. The mathematical statement and formulation of the problem are discussed below in detail.

As in [5] a row vector is said to be circular if its first and last elements are considered to be consecutive. A matrix is called row circular if its rows are circular. Given an $(n \times n)$ row circular matrix $C = \{C_{i,j}\}$, the problem is to minimize

$$Z = \max_{k=1}^{n} \sum_{i=1}^{n} C_{i,k-\pi(i)+1}$$

where $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ is a permutation of the set $N \equiv \{1, 2, \ldots, n\}$. Matrix $C$ being row circular implies that $C_{i,j \pm n} = C_{i,j} \forall i, j$. We call this problem a Cyclic Bottleneck Assignment Problem (CBAP). Cyclic refers to the row circularity of matrix $C$; bottleneck refers to the *min max* objective; and assignment refers to the problem's close affinity to the classical assignment problem that minimizes $\sum_{i=1}^{n} C_{i,\pi(i)}$.

To give the problem a different description, for a given permutation $\pi$ of the set $N$, let's define matrix $C^{\pi}$ by moving each element of row $i, i = 1, \ldots, n$, of matrix $C$ by $(\pi(i) - 1)$ positions to the right in a circular manner. More precisely, let $C^{\pi}_{i,k} = C_{i,k-\pi(i)+1} \forall\ 1 \le i \le n, 1 \le k \le n$. Since $\pi$ is a permutation, every row of matrix $C$ is rotated by a different number of columns to obtain the rotated matrix $C^{\pi}$. Furthermore, let $I_k$ denote the sum of the $k$th column of the rotated matrix $C^{\pi}$. In other words, let $I_k = \sum_{i=1}^{n} C^{\pi}_{i,k} = \sum_{i=1}^{n} C_{i,k-\pi(i)+1}$. With these new terms, the objective in our problem can be stated as min $max_{k=1}^{n}\{I_k\}$. That is, the problem is to find a permutation that minimizes the maximum column sums of the rotated matrix. Note that, with the above notation, the standard assignment problem is equivalent to min $min_{k=1}^{n} \sum_{i=1}^{n} C^{\pi}_{i,k}$.

To formulate the integer linear programming model for this problem, we define the following decision variables:

$$x_{i,j} = \begin{cases} 1 & \text{if } j = \pi(i) \\ 0 & \text{otherwise} \end{cases}$$

The model is given by

$$\begin{aligned} &\mathit{Minimize}\ Z \\ &\mathit{Subject\ to} \end{aligned} \tag{7.1}$$

$$\sum_{i=1}^{n} x_{i,j} = 1, \quad \forall \ 1 \le j \le n \tag{7.2}$$

$$\sum_{j=1}^{n} x_{i,j} = 1, \quad \forall \ 1 \le i \le n \tag{7.3}$$

$$I_k = \sum_{i=1}^{n}\sum_{j=1}^{n} C_{i,k-j+1}x_{i,j} = \sum_{i=1}^{n}\sum_{j=1}^{k} C_{i,k-j+1}x_{i,j} + \sum_{i=1}^{n}\sum_{j=1}^{n} C_{i,k-j+1+n}x_{i,j}, \quad \forall \ 1 \le k \le n \tag{7.4}$$

$$Z \ge I_k, \quad \forall \ 1 \le k \le n \tag{7.5}$$

$$x_{i,j} \in \{0,1\}, \quad \forall \ 1 \le i \le n, 1 \le j \le n \tag{7.6}$$

The objective function in Eq. 7.1 minimizes the maximum column sum of the rotated matrix $C^{\pi}$. Constraint 7.2 ensures that for each $j$ there exists an $i$ such that $j = \pi(i)$. Constraint 7.3 ensures that for each $i$ there exists a $j$ such that $j = \pi(i)$. Constraints 7.4 computes the sum of the $k$th column of the rotated matrix $C^{\pi}$. Constraint 7.5 sets the value of the objective function equal to the maximum column sum of the rotated matrix $C^{\pi}$. The CBAP is an NP-hard problem. For the proof of NP-hardness refer to the Appendix B provided in [1].

As an illustrative example, consider the following $(3 \times 3)$ row circular matrix:

$$C = \begin{bmatrix} 6 & 4 & 2 \\ 8 & 8 & 8 \\ 7 & 7 & 0 \end{bmatrix}$$

Applying the two permutations $\pi_1 = (1,2,3)$ and $\pi_2 = (1,3,2)$ of the set $\{1,2,3\}$ to matrix C yields the following rotated matrices:

$$C^{(1,2,3)} = \begin{bmatrix} 6 & 4 & 2 \\ 8 & 8 & 8 \\ 7 & 0 & 7 \end{bmatrix}, \quad C^{(1,3,2)} = \begin{bmatrix} 6 & 4 & 2 \\ 8 & 8 & 8 \\ 0 & 7 & 7 \end{bmatrix}$$

Since the column sums corresponding to permutations $\pi_1$ and $\pi_2$ are 21, 12, 17 and 14, 19, 17, respectively, the optimal solution is given by permutation $\pi_2$ yielding a minimum $Z$ value of 19. Note that due to the row circularity, we need to consider only 2 permutations in this example and $(n-1)!$ permutations in general. While the optimal solution in this example is given by $\pi_2$, the optimal solution to the standard assignment problem is given by $\pi_1$ with a minimum objective value of 12. Before closing this section, we note that if the set of constraints given in (6.5) is replaced by the single constraint $Z \ge I_1$, then we get the classical assignment problem. To see this, define

$$\hat{x}_{i,j} = \begin{cases} x_{i,j} & \text{if } j = 1 \\ x_{i,2-j+n} & \text{if } 2 \le j \le n \end{cases}$$

Now the problem $\{(6.1\text{-}6.4), Z \ge I_1, (6.6)\}$ is equivalent to $Min \sum_{i=1}^{n} \sum_{j=1}^{n} C_{i,j}\hat{x}_{i,j}$ s.t. $\sum_{i=1}^{n} \hat{x}_{i,j} = 1 \; \forall \, j, \sum_{j=1}^{n} \hat{x}_{i,j} = 1 \; \forall \, i, \hat{x}_{i,j} \in \{0, 1\}$, which is the assignment problem.

## 7.2 Probable Applications

As mentioned earlier the model stated above has applications in healthcare scheduling and supply chain management. Two specific applications of this model are described below.

### *7.2.1 Application in Healthcare*

The problem arises in surgical scheduling in a hospital setting. Surgeons operate on patients in the surgery unit. After completion of the surgery, patients are sent to the recovery unit. Assume that there are $n$ types of surgeries that need to be performed over a planning horizon of $n$ time periods (e.g. days). The goal is to develop a cyclic surgery schedule so as to minimize congestion in the recovery unit. Cyclic means that the schedule is repeated every $n$ days. Also assume that the surgery unit is open every day; that exactly one type of surgery must occur in each time period; and that patients do not stay more than $n$ days in the recovery unit. (The last assumption does not lose generality. If patients are allowed to stay more than $n$ days in the recovery unit, an equivalent problem can be formulated in which patients stay at most $n$ days.). For the case in which the identity permutation, $\pi(i) = i \; \forall \, i$, is the current schedule (or assignment, i.e. surgery type $i$ is scheduled on day $i$), $C_{i,j}$ represents the number of patients that are operated on day $i$ and are then sent to the recovery unit to remain there until the *end of day* $(i + j - 1)$. In general for a permutation $\pi$ of the set $N \equiv \{1, 2, \ldots, n\}$, the $k$th column sum, $I_k$, of the rotated matrix $C^{\pi}$ represents the number of patients remaining in the recovery unit at the end of day $k$. The maximum column sum of the rotated matrix $C^{\pi}$ represents the maximum number of patients in the recovery unit over the planning horizon. It is desirable to keep the maximum number of patients as low as possible in order to reduce the requirement of beds, nurses and other variable costs. Then, it is reasonable to ask if there exists a different permutation that can reduce the maximum number of patients. Suppose, for example, for a given permutation $\pi$, we can find another permutation $\pi'$ such that $\pi'(1) = \pi(2), \pi'(2) = \pi(1), and \; \pi'(i) = \pi(i) \; \forall \; 3 \le i \le n$ and the maximum column sum of the rotated matrix $C^{\pi'}$ is less than that of $C^{\pi}$. Then, in this case, the hospital can reduce the congestion in the recovery unit by creating a new schedule in which the positions of the surgeons that are scheduled on day 1 and day 2 are swapped and all the other surgeons keep their existing positions in the schedule. Of course, we assume that such a swap is always possible.

### 7.2.2 *Application in Supply Chain Management*

The problem arises in minimizing the space requirements in a retail store. Suppliers deliver $n$ different types of goods on $n$ different days, i.e. exactly one type of product is delivered per day. In this application, $C_{i,j}$ represents the amount of space required at the *beginning of day* $(i+j-1)$ for products delivered on day $i$. Again, we assume that suppliers deliver according to a cyclic scheduling; that the planning horizon is $n$ days; that the retail store is open every day; and that no product stays in the store for more than $n$ days. The identity permutation represents the current schedule, and $I_k$ represents the space requirement at the beginning of day $K$. Assuming that suppliers delivering on day $i$ can be swapped with those that make deliveries on day $i'$ for any $1 \le i \le n, 1 \le i' \le n, i \ne i'$, the importance of the objective $\min max_{k=1}^{n} \sum_{i=1}^{n} C_{i,k}^{\pi}$ is to minimize the maximum space requirement.

## 7.3 Cohort Intelligence (CI) Algorithm for Solving the CBAP

The CBAP presented in Sect. 7.1 is solved using the CI algorithm discussed in Chap. 2. The adaption and implementation of CI methodology for this problem is discussed below in detail.

In the context of the CI algorithm the elements of the rearrangement/permutation vector $\pi = (\pi(1), \ldots, \pi(i), \ldots, \pi(n))$ are considered the characteristics/attributes/ qualities that candidates in the cohort select and are associated with. The procedure begins with the initialization of number of cohort candidates $S$, number of variations $Y$, the permutation $\pi^s$ of every candidate $s, (s = 1, \ldots, S)$ and the convergence parameter $\varepsilon$ and maximum number of allowable learning attempts $L_{max}$.

In the cohort of $S$ candidates, every individual candidate $s, (s = 1, \ldots, S)$ randomly generates a permutation $\pi^s = (\pi(1)^s, \ldots, \pi(i)^s, \ldots, \pi(n)^s)$. Every candidate $s$ forms matrix $C^{\pi^s}$ by applying its permutation $\pi^s$ and rotating all the corresponding $n$ rows of matrix $C$ accordingly. This way, $S$ rotated matrices $\left(C^{\pi^1}, \ldots, C^{\pi^s}, \ldots, C^{\pi^S}\right)$ are formed. Next the associated vector of maximum column sums is calculated as $Z^S = \left\{Z\left(C^{\pi^1}\right), \ldots, Z\left(C^{\pi^s}\right), \ldots, Z\left(C^{\pi^S}\right)\right\}$ where $Z\left(C^{\pi^s}\right) = max_{k=1}^{n} I_k$ and $I_k = \sum_{i=1}^{n} C_{i,k-\pi(i)^s+1}^{s}$.

***Step 1.*** As a minimization problem, the probability $P^s$ of selecting a column sum $Z\left(C^{\pi^s}\right)$ of every candidate is calculated as follows:

$$P^s = \frac{1/Z\left(C^{\pi^s}\right)}{\sum_{s=1}^{S} 1/Z\left(C^{\pi^s}\right)}, \quad (s = 1, \ldots, S) \tag{7.7}$$

***Step 2.*** Every candidate $s, \quad (s = 1, \ldots, S)$ using a roulette wheel approach selects a candidate $\overbrace{s} \in (1, \ldots, S)$ in the cohort to follow, i.e. it incorporates an element from within $\pi^{\overbrace{s}}$ into its existing permutation $\pi^s$. Following a permutation means incorporating certain elements from within $\pi^{\overbrace{s}}$ into $\pi^s$. More specifically, an element $\pi(i)^{\overbrace{s}}$ from within $\pi^{\overbrace{s}}$ is selected randomly. Then the selected element $\pi(i)^{\overbrace{s}}$ is identified in $\pi^s$ along with its location. It then swaps its position with the element at the location in $\pi^{\overbrace{s}}$ corresponding to its current location in $\pi^s$. This way every candidate generates $Y$ number of permutations represented as $\Pi^{s,Y} = \left\{\pi^{s,1}, \ldots, \pi^{s,y}, \ldots, \pi^{s,Y}\right\}, \quad s = 1, \ldots, S$ and further computes the associated maximum column sums $Z\left(C^{\pi^s}\right)^Y = \left\{Z\left(C^{\pi^s}\right)^1, \ldots, Z\left(C^{\pi^s}\right)^y, \ldots, Z\left(C^{\pi^s}\right)^Y\right\}, \quad s = 1, \ldots, S$. The minimum from within $Z\left(C^{\pi^s}\right)^Y$ for every candidate $s, \quad (s = 1, \ldots, S)$ is found along with the associated permutation.

***Step 3.*** If either of the two criteria listed below is valid, accept any of the matrices from within the pool of current available rotated matrices $C^{\pi^s}, (s = 1, \ldots, S)$ as the saturated/converged matrix $C^*$ and associated permutation $\pi^*$ as the final solution and stop, else continue to *Step 1.*

(a) If the maximum number of learning attempts is exceeded.
(b) The cohort reaches a saturation state. There is no significant improvement in the elements of $Z^S$ and the difference between these elements is not very significant if further learning attempts are considered. That is, the cohort saturates to the same minimum column sum for any other number of successive learning attempts.

### *7.3.1 A Sample Illustration of the CI Algorithm for Solving the CBAP*

The CI algorithm for solving CBAP is now illustrated for the example shown in Fig. 7.1. In this example, the number of candidates is $S = 3$, the number of variations is $Y = 2$, and the number of learning attempts is $L = 1$. The initial $C$ matrix is shown in Fig. 7.1.

1. The candidates randomly generate permutations represented as $\pi^1$, $\pi^2$, and $\pi^3$ in Fig. 7.1a. Then the corresponding rotated matrices $\left(C^{\pi^1}, C^{\pi^2}, C^{\pi^3}\right)$ and associated maximum column sums $Z^3 = \left\{Z\left(C^{\pi^1}\right), Z\left(C^{\pi^2}\right), Z\left(C^{\pi^3}\right)\right\}$ are obtained.
2. The probability $P^s$, $s = 1, 2, 3$ is calculated using Eq. 7.7. The calculated probability values are presented in Fig. 7.1a.

$$C = \begin{bmatrix} 40 & 30 & 22 & 13 \\ 36 & 29 & 23 & 7 \\ 32 & 30 & 22 & 7 \\ 36 & 32 & 23 & 17 \end{bmatrix}$$

| $\pi^1 = (3,4,2,1)$ | $\pi^2 = (2,3,4,1)$ | $\pi^3 = (1,4,3,2)$ |
|---|---|---|
| $C^1 = \begin{bmatrix} 22 & 13 & 40 & 30 \\ 29 & 23 & 7 & 36 \\ 7 & 32 & 30 & 22 \\ 36 & 32 & 23 & 17 \end{bmatrix}$ | $C^2 = \begin{bmatrix} 13 & 40 & 30 & 22 \\ 23 & 7 & 36 & 29 \\ 30 & 22 & 7 & 32 \\ 36 & 32 & 23 & 17 \end{bmatrix}$ | $C^3 = \begin{bmatrix} 40 & 30 & 22 & 13 \\ 29 & 23 & 7 & 36 \\ 22 & 7 & 32 & 30 \\ 17 & 36 & 32 & 23 \end{bmatrix}$ |
| $Z\left(C^{\pi^1}\right) = 105$ | $Z\left(C^{\pi^2}\right) = 102$ | $Z\left(C^{\pi^3}\right) = 108$ |
| $P^1 = 0.333$ | $P^2 = 0.343$ | $P^3 = 0.323$ |

**(a) The candidate solutions and associated probabilities**

| | | |
|---|---|---|
| $t = 1$ | The element from $\pi^3 = (1,4,3,2)$ being followed: | 3 |
| | Its location in the permutation $\pi^1 = (3,4,2,1)$: | (1,1) |
| | The updated permutation $\pi^{1,1}$: | (2,4,3,1) |
| | Updated circular matrix $C$: | $\begin{bmatrix} 30 & 22 & 13 & 40 \\ 23 & 7 & 36 & 29 \\ 30 & 22 & 7 & 32 \\ 36 & 32 & 23 & 17 \end{bmatrix}$ |
| | Maximum column sum $Z\left(C^{\pi^{1,1}}\right)$: | 119 |
| $t = 2$ | The element from $\pi^3 = (1,4,3,2)$ being followed: | 1 |
| | Its location in the permutation $\pi^1 = (3,4,2,1)$: | (1,4) |
| | The updated permutation $\pi^{1,2}$: | $(1,4,2,3)$ |
| | Updated circular matrix $C$: | $\begin{bmatrix} 22 & 13 & 40 & 30 \\ 23 & 7 & 36 & 29 \\ 22 & 7 & 32 & 30 \\ 23 & 17 & 36 & 32 \end{bmatrix}$ |
| | Maximum column sum$Z\left(C^{\pi^{1,2}}\right)$: | 144 |
| | $Z\left(C^{*\pi^1}\right) = \min\left(Z\left(C^{\pi^{1,1}}\right), Z\left(C^{\pi^{1,2}}\right)\right)$ and associated permutation $\pi^1$: | 119 and $(2,4,3,1)$ |

**(b) Generation of Variations**

**Fig. 7.1** Illustrative example of CI solving the CBAP for a learning attempt

3. Using roulette wheel selection approach, assume that candidate 1 decides to follow candidate 3 and then generates two variations of the permutations $\Pi^{1,2} = \{\pi^{1,1}, \pi^{1,2}\}$ and associated maximum column sums $Z\left(C^{\pi^{1,1}}\right)$ and $Z\left(C^{\pi^{1,2}}\right)$ are calculated.
4. Further $Z\left(C^{\pi^{1}}\right) = \min\left(Z\left(C^{\pi^{1,1}}\right), Z\left(C^{\pi^{1,2}}\right)\right)$ and associated permutation $\pi^1 = \pi^{1,1}$ are identified.
5. In this way, candidates 2 and 3 also follow certain candidate in the cohort and find the $Z\left(C^{\pi^{2}}\right)$ and $Z\left(C^{\pi^{3}}\right)$ along with associated $\pi^2$ and $\pi^3$.

This process continues until convergence.

## 7.3.2 Numerical Experiments and Results

The CI algorithm discussed in Sect. 7.3 for solving the CBAP is coded in MATLAB 7.7.0 (R2008B). The simulations are run on a Windows platform using an Intel Core2 Quad CPU, 2.6 GHz processor speed and 4 GB memory capacity. The CI parameters such as number of candidates $S$ and number of variations $T$ are chosen to be 25 and 5, respectively. The problem size is determined by the order $n \times n$ of matrix $C$. In total, seventeen distinct cases with increasing problem size $n$ = 5 to 13, 15, 20, 25, 30, 35, 40, 45, 50 are solved. For every case, 10 instances are generated and every instance is solved 20 times using the CI method. The CI saturation/convergence plot for problem instance $n$ = 30 is presented in Fig. 7.2. This plot exhibits the self-adaptive learning behavior of every candidate in the cohort. Initially, the individual behavior/solution of every candidate in the cohort

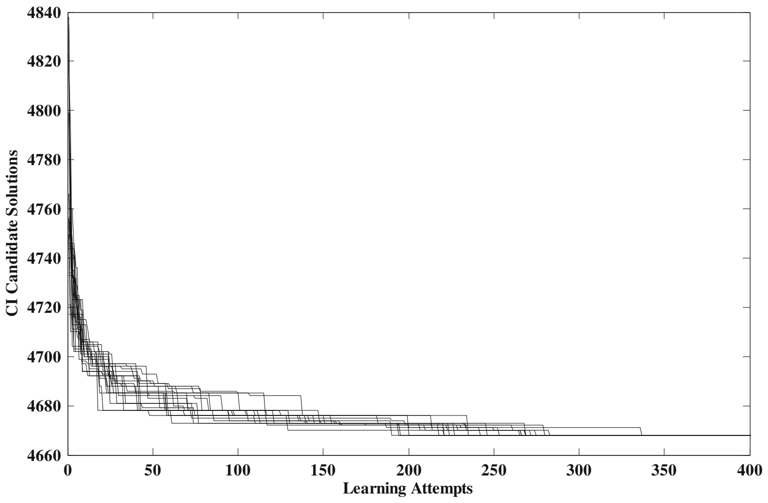

**Fig. 7.2** Saturation/convergence of the cohort

can be easily distinguished. The behavior/solution here refers to the maximum column sum $Z$ of the rotated matrix $C^{\pi}$. As every candidate adopts the qualities of other candidates to improve its own solution, the entire cohort gradually reaches a saturation stage and converges to an improved solution. It is important to mention here that the saturation associated restart procedure implemented in the original CI approach [2] which helped the candidates to explore further in the close neighborhood of their recently adopted qualities is not required.

In Table 7.1 we report our computational results obtained by solving the IP model given in Eqs. 7.1–7.6 for different values of the problem size $n$. For the solution quality, Table 7.1 shows the percentage gap between the best objective function values of solutions obtained using the LP relaxation of the model, CPLEX, the CI and MRSLS procedures. The percentage gap value between solution results of method X versus method Y is computed as $|Z_Y - Z_X| \times 100\ \%/Z_X$. These results are also summarized graphically in Fig. 7.3. First, as can be seen from columns 2 and 3 of Table 7.1, the LP relaxation of the model yields a tight lower bound that tends to improve as $n$ is increased. This is a useful finding as it allows us to assess the performance of the CI method for large problem sizes. Indeed, as is evident from Fig. 7.3a, the times taken by CPLEX to solve the problem grow exponentially large as $n$ increases. Unfortunately, we are able to report the CPU times for CPLEX only for $n$ not exceeding 13. For $n$ larger than 13, the times become prohibitively lengthy. That being said, a close examination of Table 7.1 reveals that the performance of CI method in solving CBAP is excellent both in terms of the percentage gap between the objective function values and the run times to solve the problem. For example, for $n = 13$, CPLEX takes close to 1073 s to reach an optimal solution whereas CI takes less than a second to produce a solution yielding an objective-value gap relative to CPLEX of less than 0.3 %. Also, the overall CPU time (refer to Fig. 7.3e) for CI is significantly less as compared to CPLEX. Furthermore, for comparatively smaller size cases, the solution obtained using CI method confirms with the CPLEX solution. An important observation from Table 7.1 and Fig. 7.3c, d is that similar to the percentage difference between the solution obtained using LP relaxation and CPLEX, the percentage difference between solution obtained using CI procedure and LP relaxation reduces gradually as the problem size increases. This demonstrates the noteworthy ability of CI in solving larger size problems with reasonable accuracy and also underscores its competitiveness with the CPLEX. Furthermore, the CI method could achieve the optimum solution for every case of the problem in reasonable number of function evaluations (FE). In addition, it is evident from Table 7.1 and Fig. 7.3f, g that the average number of FE is found to be increasing linearly while the standard deviation (SD) remains almost stable as the problem size increases. Since the search space increases with an increase in problem size, the number of characteristics a candidate learns from the other candidate being followed in a learning attempt do not change. This results into an increase in the number of learning attempts in order to improve their individual solution and eventually reach the saturation stage. Also, the SD presented in Table 7.1 demonstrates that the CI approach produces sufficiently robust solution for every case of the problem.

**Table 7.1** Percentage gaps and CPU times of solutions obtained using the LP relaxation, CPLEX, and CI

| Problem Size $n$ | Average % gap LP versus CPLEX | CPLEX: average CPU time (s) | % gap LP versus MRSLS | % gap IP versus MRSLS | CI method | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | % gap CI versus MRSLS | Average % gap LP versus CI | Average % gap CPLEX versus CI | CI solution SD | CPU time (s) | FE |
| 5 | 4.3165 | 0.27 | 4.5496 | 0 | 0 | 4.5496 | 0 | 0 | 0.0380 | 1620 |
| 6 | 3.3520 | 0.29 | 3.4857 | 0 | 0 | 3.4857 | 0 | 0 | 0.0824 | 3169 |
| 7 | 2.4426 | 0.41 | 2.5067 | 0 | 0 | 2.5067 | 0 | 0 | 0.1416 | 5070 |
| 8 | 1.7478 | 0.73 | 1.8423 | 0.0601 | 0.0598 | 1.7815 | 0 | 0.6597 | 0.2060 | 6988 |
| 9 | 1.3584 | 4.10 | 1.4842 | 0.1049 | 0.1046 | 1.3778 | 0 | 1.9202 | 0.3011 | 9136 |
| 10 | 0.9259 | 13.80 | 1.1638 | 0.2268 | 0.2139 | 0.9473 | 0.0120 | 3.2651 | 0.3597 | 10,750 |
| 11 | 0.8597 | 98.31 | 1.3708 | 0.4991 | 0.3912 | 0.9739 | 0.1057 | 2.9616 | 0.4433 | 12,190 |
| 12 | 0.7083 | 260.59 | 1.2766 | 0.5590 | 0.3422 | 0.9296 | 0.2146 | 3.0454 | 0.5370 | 13,601 |
| 13 | 0.5632 | 1072.93 | 1.2507 | 0.6804 | 0.3784 | 0.8675 | 0.2993 | 2.6952 | 0.6633 | 15,477 |
| 15 | – | – | 1.2879 | – | 0.4740 | 0.8040 | – | 2.8324 | 0.8314 | 17,827 |
| 20 | – | – | 0.9293 | – | 0.2772 | 0.6492 | – | 2.7274 | 1.6599 | 26,085 |
| 25 | – | – | 0.8442 | – | 0.2984 | 0.5431 | – | 2.6360 | 3.1387 | 35,435 |
| 30 | – | – | 0.6627 | – | 0.2087 | 0.4502 | – | 2.6683 | 5.0053 | 43,715 |
| 35 | – | – | 0.5657 | – | 0.1617 | 0.4031 | – | 2.5869 | 7.8426 | 51,145 |
| 40 | – | – | 0.5607 | – | 0.2177 | 0.3417 | – | 2.6962 | 11.9725 | 61,610 |
| 45 | – | – | 0.5086 | – | 0.1942 | 0.3130 | – | 2.4685 | 17.0199 | 71,835 |
| 50 | – | – | 0.4741 | – | 0.19482 | 0.2784 | – | 2.5459 | 23.9808 | 81,207 |

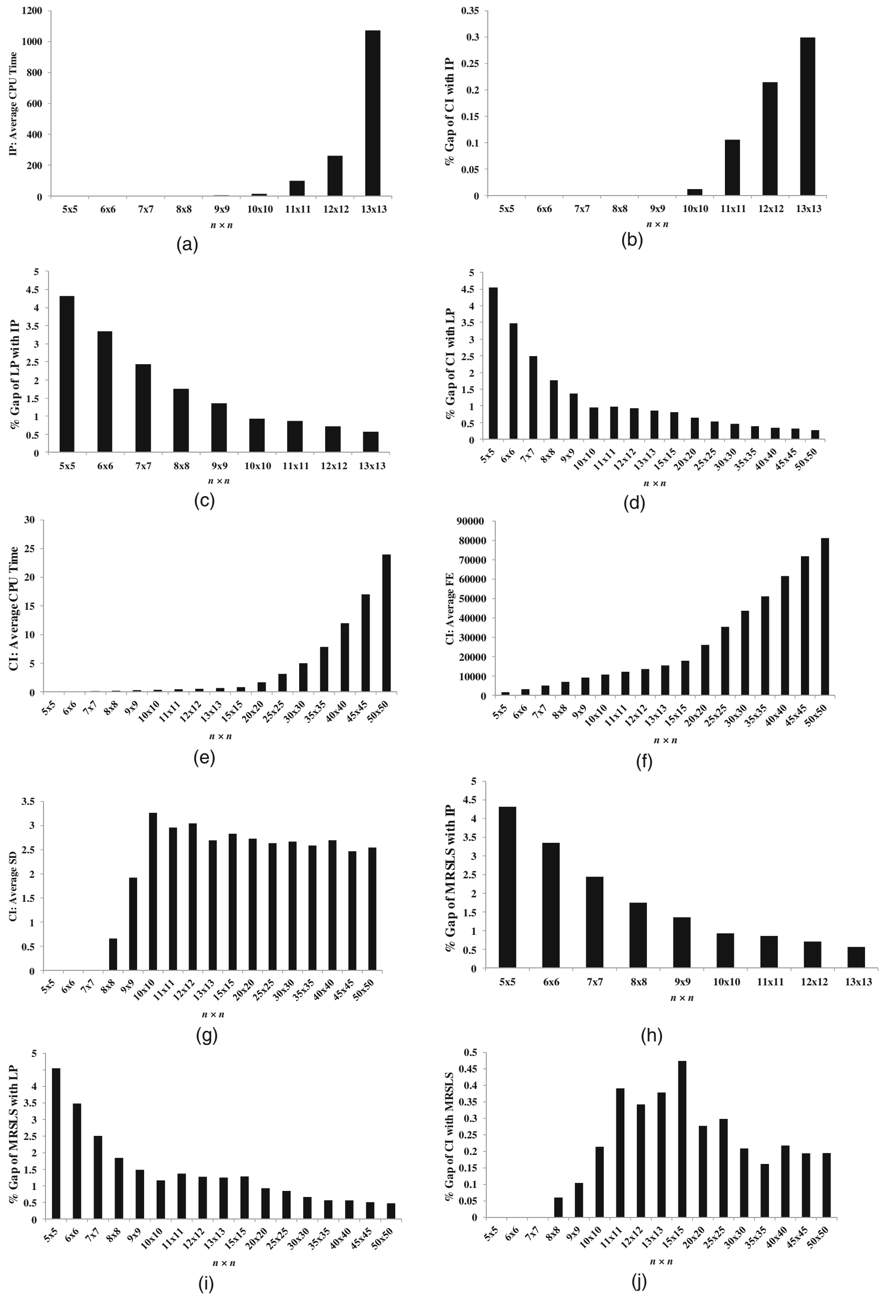

**Fig. 7.3** Illustration of the CI, CPLEX, MRSLS and LP solution comparison

In addition, we compare the performance of the CI method for solving the CBAP to that of a local search technique that moves from one solution $\pi_1$ to another neighboring solution $\pi_2$ according to some prescribed rule. The following multi-random-start local search (MRSLS) is considered. In step 1, a starting solution (or permutation of the set $\{1, 2, \ldots, n\}$) $\pi_1 = (i_1, \ldots, i_k, \mathrm{i}, \mathrm{j}, \ldots, i_n)$ is randomly generated and a value for $Z(C^{\pi_1})$ is obtained. In step 2, we use a pairwise interchange approach to generate a neighboring solution $\pi_2$ which is given by $\pi_2 = (i_1, \ldots, i_k, \mathrm{j}, \mathrm{i}, \ldots, i_n)$; that is, the two elements *i*and *j* occupying adjacent positions in the current solution are interchanged. We then calculate the corresponding $Z(C^{\pi_2})$ value. In step 3, the incumbent best solution is updated to $\pi_2$ if $Z(C^{\pi_2}) < Z(C^{\pi_1})$; otherwise $\pi_1$ is kept as the best incumbent solution found so far (ties may be broken arbitrarily). The process is continued by performing and evaluating other pairwise interchange until a stopping criteria is met. Furthermore, for every individual CBAP case considered, MRSLS is run 50 times with different initialization. Also, for a meaningful comparison, every MRSLS case is initialized to start in the neighborhood of the CI's starting point and is run for exactly the same time equal to the corresponding average CPU time the CI method takes to solve that case. The results are summarized in Table 7.1 and Fig. 7.3c, h. The results show that, while the CI method in most cases has a slight edge over MRSLS in terms of optimality gap, the two methods perform quite well in finding good solutions to the CBAP.

## 7.4 Conclusions

The emerging optimization technique of cohort intelligence (CI) is successfully applied to solve the new variant of the assignment problem, which has applications in healthcare and supply chain management. The results indicate that the accuracy of solutions to these problems obtained using CI is fairly robust and the computational time is quite reasonable. The chapter also describes the application of a multi-random-start local search (MRSLS) that can be used to solve the problem cases. The MRSLS implemented here is based on the interchange argument, a valuable technique often used in sequencing, whereby the elements of two adjacent solutions are randomly interchanged in the process of searching for better solutions. Our findings are that the two methods perform equally well in solving the CBAP, in part due the special structure of the problem.

## References

1. Kulkarni, A.J., Baki, M.F., Chaouch, B.A.: Application of the cohort-intelligence optimization method to three selected combinatorial optimization problems. Eur. J. Oper. Res. **250**(2), 427–447 (2016)
2. Kulkarni, A.J., Durugkar I.P., Kumar M.: Cohort intelligence: a self supervised learning behavior. In: Proceedings of IEEE International Conference on Systems, Man and Cybernetics, Manchester, UK, pp. 1396–1400, 13–16 Oct 2013

3. Kulkarni, A.J., Shabir, H.: Solving 0-1 Knapsack problem using cohort intelligence algorithm. Int. J. Mach. Learn. Cybernet. (2014). doi:10.1007/s13042-014-0272-y
4. Krishnasamy, G., Kulkarni, A.J., Paramesaran, R.: A hybrid approach for data clustering based on modified cohort intelligence and K-means. Expert Syst. Appl. **41**(13), 6009–6016 (2014)
5. Bartholdi, J.J., Orlin, J.B., Ratliff, H.D.: Cyclic scheduling via integer programs with circular ones. Oper. Res. **28**(5), 1074–1085 (1980)