Chapter 8
Solution to Sea Cargo Mix (SCM) Problem Using Cohort Intelligence Algorithm

Nomenclature

$\tilde{T}$	set of time periods $\{1, \ldots, t, \ldots, T\}$. Each period may represent one day, one week or one month, etc.
$\tilde{J}$	Set of ports of destinations for cargoes $\{1, \ldots, j, \ldots, J\}$
$\tilde{K}$	Set of all cargoes $\{1, \ldots, k, \ldots, K\}$ received in the planning horizon
η_k	The period that cargo k will be received at the port of origin
τ_k	The shipment due date for cargo k. Each cargo has its due date requested by shipper in its booking status
ξ_k	The port of destination for cargo k. Cargo k will be received in period η_k and will be shipped to its destination port ξ_k on or before its due date τ_k
r_{kt}	Per volume profit of cargo k which will be shipped in period t. It can be interpreted as the per volume net profit of cargo k, i.e., per volume revenue of cargo k minus its per volume delivery cost and inventory cost
E_t	Total volume of available empty containers at the port of origin in period t
$V_{t,j}$	Total available volume capacity of shipment to port j in period t
$W_{t,j}$	Maximum allowable weight capacity of shipment to port j in period t
v_k	Volume of cargo k
w_k	Weight of cargo k
x_{kt}	Binary variable, i.e., $x_{kt} = 1$, if cargo k is ready for shipment in period t; $x_{kt} = 0$, otherwise

The methodology of Cohort Intelligence (CI) [1–4] has been applied successfully applied solving combinatorial problems such as Knapsack problem, Traveling Salesman Problem and the new variant of the assignment problem (also referred to as Cyclic Bottleneck Problem (CBAP)). This chapter discusses CI solution to the Sea Cargo Mix (SCM) problem is originally proposed in [5]. The performance of CI solving the SCM is compared with the Integer Programming (IP) Solution as well as a multi-random-start local search (MRSLS) method. In addition the solution is compared with the Heuristic algorithm for MDMKP (HAM) and the Modified Heuristic algorithm for MDMKP (HAM) [5].

A.J. Kulkarni et al., *Cohort Intelligence: A Socio-inspired Optimization Method*,
Intelligent Systems Reference Library 114, DOI 10.1007/978-3-319-44254-9_8

8.1 Sea Cargo Mix Problem

As mentioned before, the Sea Cargo Mix (SCM) problem is originally proposed in [5]. The decision problem consists of choosing a sea cargo shipping schedule of accepted freight bookings over a multi-period planning horizon. The goal is to maximize profit subject to constraints such as the limited available volume capacity, weight capacity and the number of available containers at the port of origin. The mathematical formulation of this problem, which can be viewed as a multi-dimension multiple knapsack problem (MDMKP), is discussed below.

$$\textit{Maximize } Z = \sum_{1 \le k \le K} \sum_{\eta_k \le t \le \tau_k} v_k r_{kt} x_{kt} \tag{8.1}$$

Subject to

$$\sum_{k \in \tilde{K}_t} v_k x_{kt} \le E_t, \quad \forall t \in \tilde{T} \tag{8.2}$$

$$\sum_{k \in \tilde{K}_{tj}} v_k x_{kt} \le V_{tj}, \quad \forall t \in \tilde{T}, \ \forall j \in \tilde{J} \tag{8.3}$$

$$\sum_{k \in \tilde{K}_{tj}} w_k x_{kt} \le W_{tj}, \quad \forall t \in \tilde{T}, \ \forall j \in \tilde{J} \tag{8.4}$$

$$\sum_{\eta_k \le t \le \tau_k} x_{kt} \le 1, \quad \forall k \in \tilde{K} \tag{8.5}$$

$$x_{kt} \in \{0, 1\}, \quad \forall k \in \tilde{K}, \quad t\{\eta_k, \eta_k + 1, \ldots, \tau_k\} \tag{8.6}$$

where

$$\tilde{K}_t = \{k : k \in \tilde{K}, \quad \eta_k \le t \le \tau_k\}, \quad \forall t \in \tilde{T},$$
$$\tilde{K}_{tj} = \{k : k \in \tilde{K}, \quad \eta_k \le t \le \tau_k, \xi_k = j\}, \quad \forall t \in \tilde{T}, \ j \in \tilde{J}$$

The objective function (8.1) maximizes the total profit generated by all freight bookings accepted in the multi-period planning horizon T. Constraint (8.2) ensures that the demand for empty containers at the port of origin is less than or equal to the number of all available empty containers at the port of origin in each period. Constraint (8.3) ensures that the total volume of cargoes which will be carried to port j in period t is less than or equal to the total available volume capacity of shipment to port j in period t. Constraint (8.4) indicates that the total weight of cargoes which will be carried to port j in period t is less than or equal to the total available weight capacity of shipment to port j in period t. Constraint (8.5)

stipulates that each cargo may be carried in a certain period on or before its due date or refused to be carried in the time horizon T. Constraint (8.6) states that each cargo is either accepted in its entirety or turned down.

There are J destination ports and T periods in the problem, and each cargo is either to be delivered within its due date or refused to be carried in the planning horizon. Thus, the total number of knapsacks is $T \times J$. Moreover, for each knapsack, there are three constraint sets, i.e., the set associated with the number of available empty containers, amount of available volume capacity and amount of available weight capacity.

8.2 Cohort Intelligence for Solving Sea Cargo Mix (SCM) Problem

In the context of CI algorithm presented in Chap. 2, the elements of cargo assignment set $C = k_t^{\xi_k}$ formed by assigning every cargo $k, \quad k \in \{1, 2, \ldots, K\}$ to a period $t \in \{1, 2, \ldots, T\}$ being shipped to its port of destination ξ_k are considered as characteristics/attributes/qualities of the cohort candidate. The port of destination ξ_k for every cargo $k \in \{1, 2, \ldots, K\}$ is selected based on the condition below.

$$\begin{array}{ll} \xi_k = j, & if\ [K/J] \times (j-1) < k \leq [K/J] \times j, \quad for\ j = 1, 2, \ldots, J-1 \\ \xi_k = J, & if\ [K/J] \times (j-1) < k \leq K, \qquad\qquad\qquad\quad \text{otherwise} \end{array} \tag{8.7}$$

The CI algorithm begins with the initialization of number of cohort candidates S, number of variations Y the cargo assignment set C^s of every candidate s, $(s = 1, \ldots, S)$, the convergence parameter ε and maximum number of allowable learning attempts L_{max}.

In the cohort, every candidate $s, (s = 1, \ldots, S)$ randomly assigns every cargo $c_k, \quad k \in \{1, 2, \ldots, K\}$ to a period $t \in \{1, 2, \ldots, T\}$ to be shipped to destination ξ_k and forms a cargo assignment set (behavior) $C^s = k_t^{s,\xi_k}$ and associated per volume profit are calculated as $R^s = \sum_{k=1}^{K} \sum_{t=1}^{1} r_{k,t}^s$.

Step 1. (**Constraint Handling**) As a maximization problem, the probability associated with per volume profit of cargo R^s is calculated as follows:

$$p_R^s = \frac{R^s}{\sum_{s=1}^{S} R^s}, \quad (s = 1, \ldots, S) \tag{8.8}$$

There are constraints involved such as:

1. demand of empty containers $\sum_k v_{k,t}^s$ at the port of origin should be less than or equal to the number of all available empty containers E_t at the port of origin in each period $t \in \{1, 2, \ldots, T\}$

2. total volume of cargoes $\sum_k v^s_{k,j}$ which will be carried to port $j \in \{1, 2, \ldots, J\}$ in period $t \in \{1, 2, \ldots, T\}$ is less than or equal to the total available volume capacity $V_{t,j}$, and
3. total weight of cargoes $\sum_k v^s_{k,j}$ which will be carried to port $j \in \{1, 2, \ldots, J\}$ in period $t \in \{1, 2, \ldots, T\}$ is less than or equal to the corresponding total available weight capacity $W_{t,j}$.

Kulkarni and Shabir [3] propose a modified approach to the CI method for solving knapsack problems. This approach makes use of probability distributions for handling constraints. This approach is also adopted here. For every constraint type as described in 1, 2 and 3 above a probability distribution is developed (refer to Fig. 8.1) and the probability is calculated based on the following rules:

1. If $0 \leq \sum_k v^s_{k,t} \leq E_t, \ \forall t$, then based on the probability distribution presented in Fig. 8.1a $p^s_{E_t} = slope_{1,Et} \times \left(\sum_k v^s_{k,t} - E_t\right)$, else $p^s_{E_t} = slope_{1,Et} \times (0.001\ \%E_t)$.
2. If $0 \leq \sum_k v^s_{k,j} \leq V_{t,j}, \ \forall t, \forall j$, then based on the probability distribution presented in Fig. 8.1b $p^s_{V_{t,j}} = slope_{1,V_{t,j}} \times \left(\sum_k v^s_{k,j} - V_{t,j}\right)$, else $p^s_{V_{t,j}} = slope_{1,V_{tj}} \times \left(0.001\ \%V_{t,j}\right)$.
3. If $0 \leq \sum_k w^s_{k,j} \leq W_{t,j}, \ \forall t, \forall j$, then based on the probability distribution presented in Fig. 8.1c $p^s_{W_{t,j}} = slope_{1,W_{t,j}} \times \left(\sum_k w^s_{k,j} - W_{t,j}\right)$, else $p^s_{W_{tj}} = slope_{1,W_{tj}} \times \left(0.001\ \%W_{t,j}\right)$.

As represented in Fig. 8.1, the $slope_{1,Et}$, $slope_{1,V_{t,j}}$ and $slope_{1,W_{t,j}}$ represent the slope of lines going through points $((0,1),(E_t,0))$, $\left((0,1),\left(V_{tj},0\right)\right)$ and $\left((0,1),\left(W_{tj},0\right)\right)$, respectively. The overall (total) probability of selecting candidates to follow candidate $s, (s = 1, \ldots, S)$ is calculated as follows:

$$p^s = \left(p^s_R + \sum\nolimits_t p^s_{E_t} + \sum\nolimits_t \sum\nolimits_j p^s_{V_{t,j}} + \sum\nolimits_t \sum\nolimits_j p^s_{W_{t,j}}\right) \tag{8.9}$$

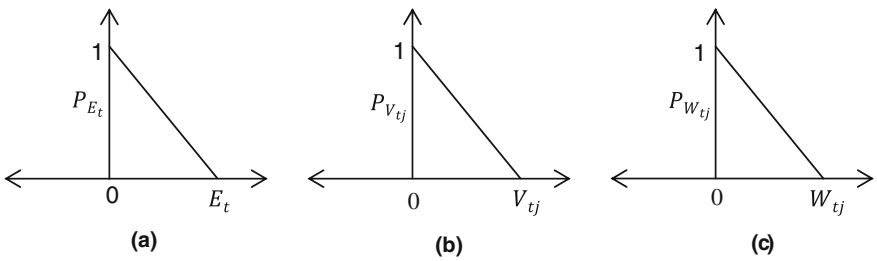

Fig. 8.1 Probability distributions for constraint handling

It is clear from the above rules for probability calculation that the candidate's behavior/solution/cargo assignment with better objective and constraint values closer to the boundaries will have higher probability of being followed.

Step 2. Every candidate generates Y new variations of the cargo assignment using two steps, which we refer to as 'learning from others' and 'introspection', as follows:

1. **Learning from others**: Every candidate $s, \quad (s = 1, \ldots, S)$ using roulette wheel approach [1–4] selects a candidate $\overset{\frown}{s} \in (1, \ldots, S)$ (not known in advance) in the cohort to follow, i.e. it incorporates an element from within $c^{\overset{\frown}{s}}$ into its existing cargo assignment c^s. More specifically, a quality from within $c^{\overset{\frown}{s}}$ is selected randomly. Then the selected element is identified in c^s along with its location. It then swaps its position with the element at the location in c^s corresponding to its current location in $c^{\overset{\frown}{s}}$. This way every candidate $s, \quad (s = 1, \ldots, S)$ generates $Y/2$ cargo assignments.
2. **Introspection**: In addition, every candidate $s, \quad (s = 1, \ldots, S)$ randomly selects an element from within its one of the periods $t, \quad (t = 1, \ldots, T)$ and relocates it to another period. This way every candidate $s, \quad (s = 1, \ldots, S)$ generates further $Y/2$ cargo assignments.

This way every candidate forms a total of Y new variations $\mathrm{C}^{s,Y} = \left\{\mathrm{c}^{s,1}, \ldots, \mathrm{c}^{s,y}, \ldots, \mathrm{c}^{s,Y}\right\}$ and computes associated per volume profit and constraint functions.

Step 3. As discussed in Step 1, every candidate $s, \quad (s = 1, \ldots, S)$ calculates its corresponding probability vector $P^{s,Y} = \left\{p^{s,1}, \ldots, p^{s,y}, \ldots, p^{s,Y}\right\}$. Furthermore, based on the feasibility-based rules shown below, the candidate accepts or rejects the solution associated with the maximum total probability value, i.e. $max\left\{p^{s,1}, \ldots, p^{s,y}, \ldots, p^{s,Y}\right\}$

The feasibility-based rules are as follows:
Accept the current behavior/solution if

1. The cargo assignment in the previous learning attempt is feasible and current behavior/cargo assignment is also feasible with improved per volume profit
2. The cargo assignment in the previous learning attempt is infeasible and the current behavior/cargo assignment is feasible
3. The cargo assignment in the previous learning attempt is infeasible and the current behavior/cargo assignment is also infeasible with the maximum total probability value improved;

Otherwise, reject the current behavior/solution and retain the previous one if

1. The cargo assignment in the previous learning attempt is feasible and current cargo assignment is infeasible
2. The cargo assignment in the previous learning attempt is feasible and the current cargo assignment is also feasible with worse per volume profit
3. The cargo assignment in the previous learning attempt as well as current learning attempt are infeasible and the total probability value is lesser than the previous learning attempt.

After the completion of step 3, a cohort with S updated cargo assignments $\{c^1, \ldots, c^s, \ldots, c^S\}$ is now available.

Step 4. If either of the two criteria listed below is valid, accept the best possible cargo assignment from within the available $\{c^1, \ldots, c^s, \ldots, c^S\}$ in the cohort as the final solution c^* and stop, else continue to Step 1

(a) If maximum number of learning attempts exceeded or
(b) The cohort is saturated, i.e. if cohort candidates saturate to the same cargo assignment for any other number of successive learning attempts.

8.3 Numerical Experiments and Results

The following notation is used to describe the results of our numerical experiments:

N_v	Number of decision variable in the problem
N_c	Number of constraints in the problem
N_{in}	Number of tested instances
U	Upper bound
I	Integer programming solution (branch-and-bound method)
L	LP relaxation
H	Heuristic algorithm for MDMKP (HAM) (refer to [5])
M	Modified heuristic algorithm for MDMKP (MHA) (refer to [5])
CI	Cohort intelligence (CI) method
MRSLS	Multi-Random-Start Local Search
g_{XZ}	Average percentage gap between the best objective value of the solutions obtained using methods X and Z
$g_{XZ}^{\sim}$	Average percentage gap between the average objective value of the solutions obtained using methods X and Z
$g_{XZ}^{\wedge}$	Worst percentage gap between the worst objective value of the solutions obtained using methods X and Z

(continued)

(continued)

t_X	Average computational time (in seconds) of algorithm X
$S_{t_{CI}}$	Standard deviation of CPU time for CI method
S_{XZ}	Standard deviation of percentage gap between objective value of the solutions obtained using methods X and Z

The CI approach for solving the SCM Problem discussed in Sect. 8.1 is coded in MATLAB 7.7.0 (R2008B). The simulations are run on a Windows platform with an Intel Core2 Quad CPU, 2.6 GHz processor speed and 4 GB memory capacity. For this model, we solve 18 distinct cases. These cases, which are originally proposed in [5], are presented in Tables 8.1, 8.2 and 8.3. For every case, 10 instances are generated and every instance is solved 10 times using the CI method. The instances are generated as suggested in [5]. The per volume profit $r_{k,t}$ for cargo k, $(k = 1, \ldots, K)$ shipped in period t, $(t = 1, \ldots, T)$ are uniformly generated in the interval $[0.01, 1.01]$. The volume v_k and weight w_k of every cargo c_k, $(k = 1, \ldots, K)$ are uniformly generated from the interval $[100, 200]$. The number of all available empty containers E_t at the port of origin in each period $t \in \{1, 2, \ldots, T\}$ are uniformly generated from the interval $[100 \times (K/T), 200 \times (K/T)]$, and the total volume $V_{t,j}$ and weight $W_{t,j}$ of cargoes which are carried to port $j \in \{1, 2, \ldots, J\}$ in period $t \in \{1, 2, \ldots, T\}$ are uniformly generated from the interval $[100 \times (K/T), 200 \times (K/T)]$.

The CI parameters such as number of candidates S and number of variations Y are chosen to be 3 and 15, respectively. The CI saturation/convergence plot for one problem instance given by $(T, J, K) = (4, 13, 5479)$ is presented in Fig. 8.2. The plot exhibits the self-adaptive learning behavior of every candidate in the cohort. Initially, the distinct behavior/solution of every individual candidate in the cohort can be easily distinguished. The behavior/solution here refers to the total profit generated by all freight bookings accepted in the multi-period planning horizon T. As each candidate adopts the qualities of other candidates to improve its own behavior/solution, the behavior of the entire cohort saturates/converges to an improved solution.

The best and average CI solution for the objective function value for every case is compared with the associated upper bound (UB) solution achieved by solving the LP relaxation of the problem, and the integer programming (IP) solution. In addition, the solution is compared to the solution of the LP relaxation, and the problem-specific heuristic algorithm for MDMKP (HAM) and the modified heuristic algorithm for MDMKP (MHA) developed in [5]. The numerical results are presented in Tables 8.1, 8.2 and 8.3 along with the graphical illustration in Fig. 8.3. It is important to mention here that IP is not able to solve large-scale SCM problems.

Table 8.1 Results for small scale test problems

T, J, K	N_v	N_c	N_{in}	IP	LP (L)	HAM			MHA		
				CPU time (s) t_I	CPU time (s) t_L	g_{IH}	g_{LH}	CPU time (s) t_I	CPU time (s) t_L	g_{IH}	g_{LH}
3, 5, 41	123	74	10	0.106	0.028	1.59	2.88	0.001	1.55	2.71	0.015
3, 6, 47	141	86	10	0.274	0.047	1.26	2.67	0.002	1.03	2.42	0.023
4, 3, 64	256	92	10	0.480	0.183	0.87	2.32	0.011	0.67	1.53	0.056
2, 4, 132	264	150	10	0.148	0.391	0.68	2.03	0.016	0.51	1.18	0.093
3, 3, 91	273	112	10	0.257	0.289	0.93	1.98	0.008	0.78	1.79	0.046
2, 3, 143	286	157	10	0.096	0.485	0.46	1.03	0.014	0.28	0.68	0.078
CI Performance											
Best sol % gap	Avg sol % gap	Worst sol % gap	Best sol % gap	Avg sol % gap	Worst Sol % gap	CPU time (s)	SD (CPU time)	SD	SD		
0.5375	1.3188	2.3818	0.0237	0.2452	1.8782	0.070	0.036	0.691	0.687		
0.3902	1.0102	1.6991	0.0709	0.6931	1.3842	0.093	0.027	0.456	0.454		
0.5466	0.9958	1.5231	0.0616	0.5131	1.0430	0.103	0.025	0.351	0.349		
0.4813	1.0906	1.6339	0.3960	1.0059	1.5496	0.098	0.041	0.408	0.408		
0.3758	0.9250	1.6017	0.1720	0.8474	1.4006	0.130	0.046	0.463	0.462		
0.1827	0.8045	1.3027	0.1424	0.7644	1.2629	0.094	0.037	0.379	0.378		

It is evident from the results in Tables 8.1, 8.2, 8.3 and the plots given in Fig. 8.3a, d that, for small scale SCM problems, the CI method produces a solution that is fairly close to the IP and UB solution. The gap gradually increases as the problem size grows; however, observe that the worst gap between the best CI solution and corresponding IP (g_{ICI}) and UB solution (g_{ICI}) is within 1.0459 % of the reported IP solution and 4.0405 % of the reported UB solution, respectively. Similarly, the worst gap between the average CI solution and corresponding IP $\left(g^{\sim}_{ICI}\right)$ and UB solution $\left(g^{\sim}_{ICI}\right)$ is within 2.2682 % of the reported IP solution and 5.5827 % of the reported UB solution, respectively. Also, the percent gap between the worst CI solution and corresponding IP solution $\left(g^{\wedge}_{ICI}\right)$ is within 3.0198 % of the reported IP solution. The corresponding UB solution $\left(g^{\wedge}_{ICI}\right)$ is within 7.1465 % of the reported UB solution.

Furthermore, as shown in Tables 8.1, 8.2, 8.3 and Fig. 8.3i, j, even though the standard deviation (SD) of the percent gap between the CI solution and the corresponding IP (S_{ICI}) and UB solution (S_{ICI}) increases with the problem size, the

Table 8.2 Results for medium scale test problems

T, J, K	N_v	N_c	N_{in}	IP	LP (L)	HAM			MHA		
				CPU time (s) t_I	CPU time (s) t_L	g_{IH}	g_{LH}	CPU time (s) t_I	CPU time (s) t_L	g_{IH}	g_{LH}
3, 4, 900	2700	927	10	0.722	419.1	3.92	1.12	1.72	6.17	3, 4, 900	2700
4, 8, 965	3860	1033	10	1.833	1264.3	2.66	2.58	1.16	14.37	4, 8, 965	3860
4, 25, 1000	4000	1204	10	1.240	2012.5	2.05	8.12	0.86	49.66	4, 25, 1000	4000
2, 3, 2871	5742	2885	10	1.227	5872.2	1.35	5.11	0.75	28.29	2, 3, 2871	5742
2, 3, 3876	7752	3890	10	1.887	199966	0.56	5.97	0.32	55.41	2, 3, 3876	7752
5, 37, 1954	9770	2329	10	4.151	12306.1	1.83	57.67	1.26	321.02	5, 37, 1954	9770

CI Performance									
Best sol % gap	Avg sol % gap	Worst sol % gap	Best sol % gap	Avg sol % gap	Worst Sol % gap	CPU time (s)	SD (CPU time)	SD	SD
1.0516	2.2738	3.0253	1.0459	2.2682	3.0198	0.734	0.271	0.650	0.650
0.9356	1.3321	1.8138	0.9284	1.3250	1.8067	1.382	0.609	0.299	0.299
0.0859	0.8452	1.6848	0.0794	0.8387	1.6785	0.185	0.225	0.611	0.611
1.0009	1.7398	2.2239	0.9968	1.7357	2.2199	2.931	0.798	0.380	0.380
0.5979	1.1463	1.5484	0.5944	1.1428	1.5449	2.488	0.586	0.303	0.303
0.4617	1.4282	2.5431	0.4576	1.4241	2.5390	1.305	0.425	0.917	0.917

worst SD is 0.917. Moreover, Table 8.1, 8.2, 8.3 and Fig. 8.3f also show that the SD $(S_{t_{CI}})$ of CPU time for solving small- and medium-scale problems is within 0.046 and 0.708, respectively. For large-scale problems it is within 4.940. This is because the search space increases with an increase in problem size.

For every candidate the number of characteristics to be learnt in a learning attempt from the candidate that is being followed does not change. This results into different number of learning attempts to improve their individual behavior/solution and to eventually reach the saturation/convergence state. However, it is important to mention here that the overall SD obtained by solving the entire problem set is quite reasonable which lends support to the robustness of the algorithm.

Also, the percent gap between the worst CI solution and corresponding IP solution $\left(g^{\wedge}_{ICI}\right)$ is within 3.0198 % of the reported IP solution. The corresponding UB solution $\left(g^{\wedge}_{UCI}\right)$ is within 7.1465 % of the reported UB solution. This

Table 8.3 Results for large scale test problems

T, J, K	N_v	N_c	N_{in}	HAM	MHA	CI Performance					
				CPU time (s) t_H	CPU time (s) t_M	Best sol % gap g_{UCI}	Avg sol % gap $g_{\widetilde{UCI}}$	Worst sol % gap $g_{\widehat{UCI}}$	CPU time (s) t_{CI}	SD (CPU time)	SD S_{UCI}
9, 47, 1521	13689	2376	10	80.5	447.1	3.3822	4.6675	5.8294	6.7303	0.344	0.707
3, 4, 6576	19728	6603	10	53.2	293.1	3.8078	5.2705	6.6293	61.6805	4.940	0.726
4, 5, 5286	21144	5330	10	54.5	277.7	3.6931	5.1307	6.5576	35.6223	3.819	0.835
4, 13, 5479	21916	5587	10	125.9	785.9	3.8556	5.5827	7.1253	27.2433	2.885	0.898
5, 8, 4954	24770	5039	10	89.9	454.2	4.0504	5.5827	7.0751	41.9530	3.514	0.766
8, 26, 3249	25992	3673	10	178.3	982.9	3.6647	5.2144	7.1465	28.5853	1.212	0.892

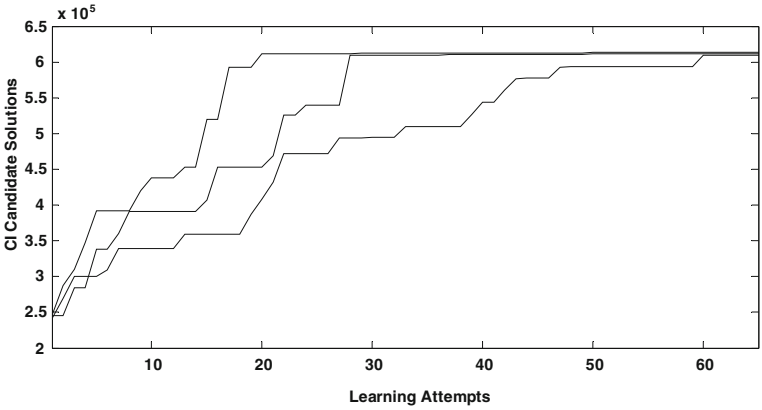

Fig. 8.2 Saturation/convergence of the cohort for instance of the SCM problem

demonstrates that, even though the magnitude of S_{ICI}, S_{UCI} and $S_{t_{CI}}$ increases with increase in problem size, CI is able to produce solutions with reasonable accuracy for every case of the problem. In addition, the CI method achieves the optimum solution for medium- and large-scale problems in significantly less CPU time (refer to Fig. 8.3h). This demonstrates the ability of CI in solving large problems efficiently and highlights its competitiveness with the IP approach as well as the heuristics HAM and MHA discussed in [5].

In addition to the above, CI's performance is also compared to the performance of a multi-random-start local search (MRSLS) that is used to solve the Sea Cargo Mix problem. The proposed MRSLS follows a similar pairwise interchange approach that we use for the CBAP discussed in Chap. 7. For each of the problem instances suggested in [5], a solution is first constructed. Then a pairwise interchange approach is used in every successive learning attempt where two time periods are selected randomly. Next a set of containers associated with each period is randomly chosen and then the positions of these two sets are interchanged (swapped). The MRSLS for every individual case of the SCM problem is run 50 times with different initializations. 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 acceptance of the resulting solution in every learning attempt depends on following feasibility-based rules (see [6] for a detailed discussion): (1) if the existing solution is infeasible and the resulting solution has improved constraint violation, then the solution is accepted, (2) If the existing

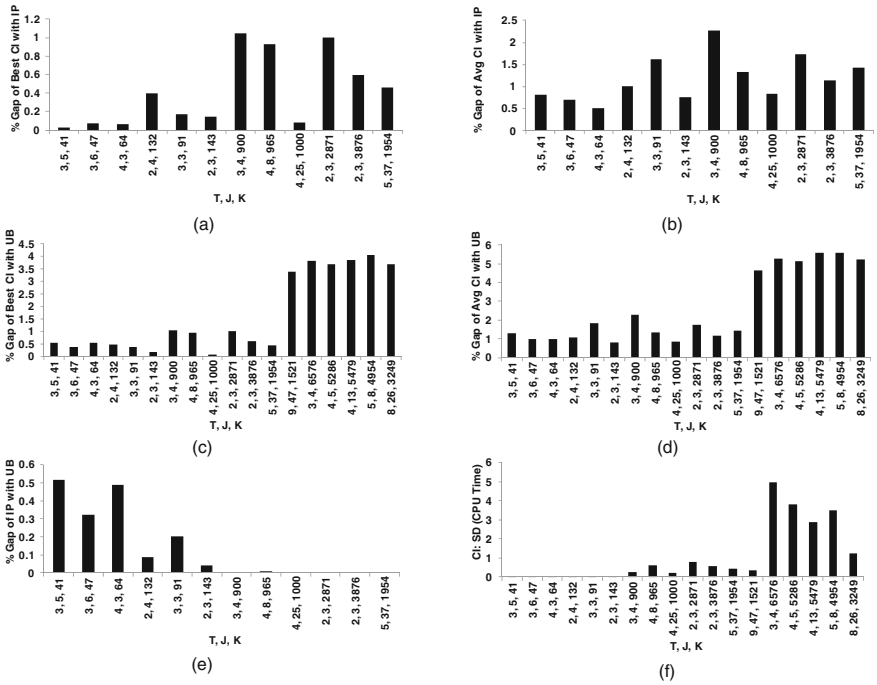

Fig. 8.3 Illustration of CI, IP, MRSLS and UB solution comparison

solution is infeasible and the resulting solution is feasible, then the solution is accepted, (3) if the existing solution is feasible and the resulting solution is also feasible yielding an improved objective function value Z, then the solution is accepted. If any of these conditions are not satisfied then the existing solution is retained and the resulting solution is discarded.

It is important to mention here that of the 50 MRSLS runs related to the SCM problems under study, only a few of the solutions obtained are feasible. Most of solutions are outside the feasible region. This is because for every MRSLS run a starting solution is randomly chosen and this solution can be infeasible. Furthermore, the MRSLS may not be able to discover a feasible solution during the entire run. Therefore, only the best of the feasible solutions are considered for meaningful comparison with the CI approach. From Tables 8.4, 8.5 and 8.6 as well as Fig. 8.3a, k it is clear that the rate of increase of the percentage gap between the solution obtained using MRSLS and that obtained using CPLEX is significantly more when compared to the rate of percentage gap increase between CPLEX and

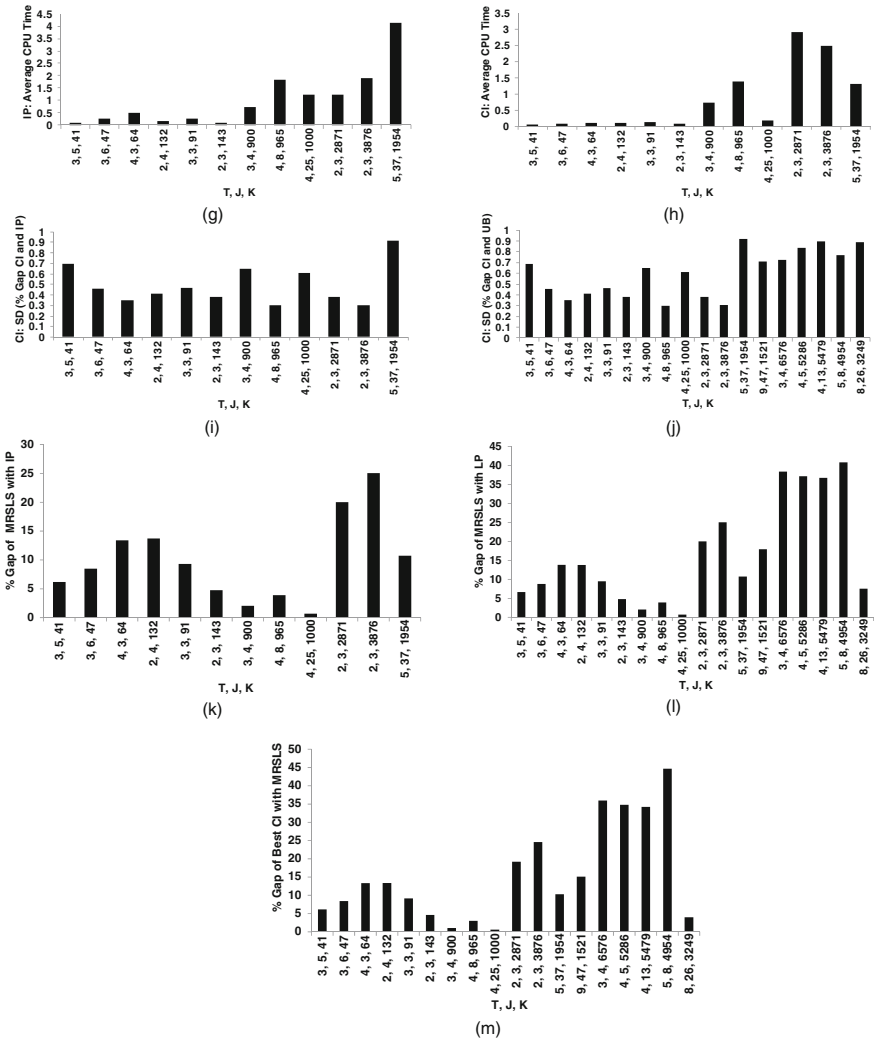

Fig. 8.3 (continued)

CI. In addition, the percentage gap between the solution obtained using MRSLS and LP relaxation for each case is also considerably larger as compared to that of CI versus LP relaxation. In short, for the Sea Cargo Mix problem, CI achieves better performance against the MRSLS implemented for this model, especially when the problem size is large.

Table 8.4 MRSLS results comparison for small scale test problems

T, J, K	Best sol % gap $g_{MRSLSLP}$	Best sol % gap $g_{MRSLSLP}$	Best sol % gap $g_{MRSLSLP}$
3, 5, 41	6.6548	6.1737	6.1516
3, 6, 47	8.8023	8.5057	8.4391
4, 3, 64	13.8332	13.4053	13.3487
2, 4, 132	13.7926	13.7174	13.3799
3, 3, 91	9.5009	9.3181	9.1595
2, 3, 143	4.8001	4.7614	4.6253

Table 8.5 MRSLS results comparison results for medium scale test problems

T, J, K	Best sol % gap $g_{MRSLSLP}$	Best sol % gap $g_{MRSLSIP}$	Best sol % gap $g_{MRSLSIP}$
3, 4, 900	2.0691	2.0635	1.0303
4, 8, 965	3.9044	3.8974	3.0055
4, 25, 1000	0.7014	0.6951	0.6178
2, 3, 2871	20.0099	20.0065	19.2088
2, 3, 3876	25.0458	25.0432	24.5916
5, 37, 1954	10.7262	10.7223	10.3139

Table 8.6 MRSLS results comparison results for large scale test problems

T, J, K	Best sol % gap $g_{MRSLSLP}$	Best sol % gap $g_{MRSLSIP}$	Best sol % gap $g_{MRSLSIP}$
9, 47, 1521	17.9624	–	15.1393
3, 4, 6576	38.3933	–	35.9608
4, 5, 5286	37.1910	–	34.7788
4, 13, 5479	36.7655	–	34.2314
5, 8, 4954	40.8499	–	44.6732
8, 26, 3249	7.5249	–	4.0031

8.4 Conclusions

The emerging optimization technique of cohort intelligence (CI) is successfully applied to solve a complex combinatorial problem such as the sea cargo mix problem. For the problem a specific CI algorithm is developed. 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 MRSLS that can be used to solve several cases of the problem.

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 performance of the CI is clearly superior to that of IP, HAM and MHA as well as the MRSLS for most of the problem instances that have been solved.

In agreement with the no-free-lunch theorem [7], any algorithm may not be directly applicable to solve all the problem types unless it can be enhanced by incorporating some useful techniques or heuristics. The CI method may also benefit from certain performance-enhancing techniques when it is applied to different classes of problems. A mechanism to solve multi-objective problems is currently being developed, which can prove helpful in transforming the model's constraints into objectives/criteria (see [7] for new development in this area). This can help reduce the dependency on the quality of the candidates' initial guess.

References

1. 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
2. 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. (2015)
3. Kulkarni, A.J., Shabir, H.: Solving 0-1 Knapsack Problem using cohort intelligence algorithm. Int. J. Mach. Learn. Cybern. (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. Ang, J.S.K., Cao, C., Ye, H.Q.: Model and algorithms for multi-period sea cargo mix problem. Eur. J. Oper. Res. **180**(3), 1381–1393 (2007)
6. Deb, K.: An efficient constraint handling method for genetic algorithms. Comput. Methods Appl. Mech. Eng. **186**(2–4), 311–338 (2000)
7. Patankar, N.S., Kulkarni, A.J., Tai, K., Ghate, T.D., Parvate, A.R.: Multi-criteria probability collectives. Int. J. Bio-Inspired Comput. **6**(6), 369–383 (2014)