International Journal of Computational Intelligence Systems, Vol. 7, No. 3 (June 2014), 515-523

An Immune Evolutionary Approach for the Label Printing Problem

Yi-Chih Hsieh*
Department of Industrial Management, National Formosa University
Huwei, Yunlin 632, Taiwan

Peng-Sheng You
Department of Marketing and Logistics/Transportation, National ChiaYi University
Chia-Yi 600, Taiwan

Received 26 August 2012

Accepted 26 November 2013

Abstract

In this short paper, the label printing problem (LPP) is investigated. The LPP can be formulated as a nonlinear integer programming problem and it aims to minimize the total wastage of labels under a fixed number of templates with subject to the minimal required quantities of various labels. Since the LPP is NP-hard and its feasible region is pretty large, it is usually difficult to solve. In this short paper, based upon a novel coding scheme, we apply an effective immune based evolutionary algorithm (IA) to solve the LPP. Numerical results show that the IA performs well for all test problems. Moreover, some best solutions by the applied IA are superior to the best well known solutions in the literature.

Keywords: label printing, immune algorithm, optimization, application.

1. Introduction

Printing is a direct or indirect reproducing approach to copy figures and/or characters in various materials, e.g., papers, plastics and cloths. Printing problem has several applications in real-world industries. There are two typical printing problems in practice, namely, the cover printing problem (CPP) and the label printing problem (LPP). The CPP aims to minimize the printing cost with subject to the minimal required prints of various book covers, while LPP aims to minimize the total wastage of labels under a fixed number of templates subject to the minimal required quantities of distinct labels. The former can be formulated as a mixed integer programming problem, and the latter can be formulated as a nonlinear integer programming problem. Since both problems are NP-hard problems and their feasible regions are pretty large, they all are difficult to solve.

To obtain higher profits in printing industry, enterprises usually have to reduce two major costs in production, including: (i) costs of materials, usually papers or cloths, and (ii) costs of plates/templates for both CPP and LPP. In a printing factory, plates/templates are usually made before the printing process. Due to the size of plate/template, the number of covers/labels allocated to each plate/template is fixed or limited. Therefore, for both printing problems, one has to simultaneously determine: (i) the number of plates/templates, (ii) their corresponding compositions, and (iii) the number of materials used.

In this paper, we confine our analysis to the LPP. The LPP has been previously investigated by Yiu et al.[1].

* Corresponding author: yhsieh@nfu.edu.tw

As mentioned above, LPP aims to minimize the total wastage of labels under a fixed number of templates subject to the minimal required quantities of distinct labels. To avoid manual separation and collection before packing labels, Yiu et al.[1] assumed that the allocation of each type of label is confined to one template only. The LPP is then formulated as a nonlinear integer programming problem (NIPP). Yiu et al.[1] presented a two-level approach to solve the LPP. The outer level is to find the types of label for template by using a simulated annealing (SA) proposed by Kirkpatrick et al.[2], while the inner level is to determine the quantity of labels for each template. In addition, several researchers have devoted to solving NIPP by the branch-and-bound method, e.g., Sandgren[3], Borchers and Mitchell[4], Fletcher and Leyffer[5], Leyffer[6]. Though the branch-and-bound approach can obtain the optimal solution for NIPP, it is time-consuming especially when the problem size is larger.

This paper aims to investigate the LPP and an effective immune based algorithm (IA) is applied to solve the problem. This paper is organized as follows. In Section 2, we will present notations, assumptions and the mathematical formulation of the LPP and then give an example to illustrate. In Section 3, we will show the main steps of IA for the LPP. Numerical results of benchmark problems by IA approach are reported and discussed in Section 4. Short conclusions are summarized in Section 5.

2. Label Printing Problem (LPP)

2.1. *Notations.*

Following Yiu et al.[1], the notations are listed below.

n	the number of templates.
m	the number of types of labels, $m \geq n$.
M	the maximal number of labels in a template.
Q_i	the quantity needed of label i, $1 \leq i \leq m$.
Q_{m+1}	$\equiv 0$, the number of empty label.
k_i	the number of prints for the ith template, $1 \leq i \leq m$.
P_{ij}	the number of the jth label assigned to the ith template, $1 \leq i \leq n$, $1 \leq j \leq m$.
$P_{i,m+1}$	the number of empty labels assigned to the ith template, $1 \leq i \leq n$.
y_{ij}	=1 if the jth label is assigned to the ith template, otherwise 0.
δ	the penalty of each redundant label.
$V(n,m)$	the wastage (%) for n templates and m distinct labels.

2.2. *Assumptions* (Yiu et al.[1])

(a) The quality, size, density, and even color of each template are identical.
(b) The sizes of prints of labels are identical.
(c) Empty labels are allowed in a template to reduce the redundancy of labels.
(d) Each template is divided into M equal areas to print labels.
(e) To avoid manual separation and aggregation before packing, we confine the allocation of each type of label to one template only.
(f) The objective is to minimize the total wastage of labels under a fixed number of templates.

2.3. *Mathematical Programming Model*

$$Min \quad V(n,m) = \sum_{j=1}^{m}\sum_{i=1}^{n} P_{ij}k_i - \sum_{j=1}^{m} Q_j + \delta \sum_{i=1}^{n} P_{i,m+1}k_i \qquad (1)$$

$$\text{st} \quad \sum_{j=1}^{m+1} P_{ij} = M, \ 1 \leq i \leq n \qquad (2)$$

$$P_{ij} \leq M y_{ij}, \ 1 \leq i \leq n, \ 1 \leq j \leq m \qquad (3)$$

$$\sum_{i=1}^{n} P_{ij}k_i \geq Q_j, \ 1 \leq j \leq m \qquad (4)$$

$$\sum_{i=1}^{n} y_{ij} = 1, \ 1 \leq j \leq m \qquad (5)$$

$$y_{ij} \in \{0,1\}, \ P_{ij}, k_i \in \{0,1,2,\ldots\}, \ 1 \leq i \leq n, \ 1 \leq j \leq m \qquad (6)$$

The objective function (1) of the model[1] is to minimize the total wastage of labels under a fixed number of templates subject to the minimal required quantities of various labels. Eq. (2) implies that the total number of labels, including empty labels, is M for each template. Constraint (3) is to confine the upper bound of number of label j in template i. Constraint (4) is to satisfy the required quantity of each label j. Eq. (5) shows that the allocation of each type of label j is confined to one template only. Constraint (6) shows the ranges of decision variables.

Example 1. Suppose that there are four distinct labels, namely, 1, 2, 3 and 4, of copies 2000, 4000, 8000, 10000 to be produced, and the maximal number of labels in a template is 12(=M). The allocation of templates in Fig. 1(a) requires 2 templates and the total

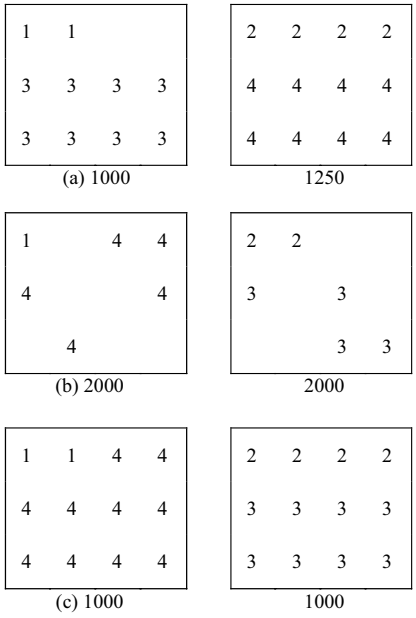

Fig. 1. Three various allocations of templates for Example 1

number of prints is 2250 (=1000+1250), the allocation of templates in Fig. 1(b) requires 2 templates and the total number of prints is 4000 (=2000+2000), and the allocation of templates in Fig. 1(c) requires 2 templates and the total number of prints is 2000 (=1000+1000). Clearly, the allocation in Fig. 1(c) is better than those in Fig. 1(a) and Fig. 1(b).

3. Immune Based Algorithm

3.1. *Immune System* (Hsieh and You[7])

As known, the natural immune system of all animals is a very complicated system to defense against pathogenic organisms. As known, the basic components in the immune system are lymphocytes and antibodies[8], and a two-tier line of defense in the immune system[9] includes:

(i) Innate immune system:
The cells in the innate immune system are immediately available to combat against a variety of new antigens which have not appeared previously.

(ii) Adaptive immune system:
The cells in the adaptive system can develop an immune memory so that they can recognize the same antigenic stimuli when they appear again to the organism.

There are two main types of lymphocytes, namely, B-lymphocytes (B-cells) and T-lymphocytes (C-cells). B-cells and T-cells carry surface receptor molecules capable of recognizing antigens. As know, B-cells produced by the bone marrow possess a distinct chemical structure and can be programmed to make only one antibody that is placed on the outer surface of the lymphocyte to act as a receptor. The antigens will only bind to these receptors with which it makes a good fit (Huang[10]).

The main task of the immune system is to discriminate and perish the intruders of the organism so that it must has the capability of both self discrimination and non-self discrimination. As mentioned above, various antibodies can be produced and then they can recognize the specific antigens. The portion of antigen recognized by antibody is called epitope which acts as an antigen determinant. Every type of antibody has its own specific antigen determinant which is called idiotype. Moreover, in order to produce enough specific effector cells to against an infection, activated lymphocyte has to proliferate and then differentiate into these effector cells. This process is called clonal selection[11] and it follows the genetic operations such that a large clone of plasma cell is formed. Therefore, the antibodies can be secreted and ready to bind antigens.

According to the above facts, based on the clonal selection theory, Jerne[12] proposed an idiotype network hypothesis. In his hypothesis, some types of recognizing sets are activated by some antigens and produce an antibody which will then activate other types of recognizing sets. By this way, the activation is propagated through the entire network of recognizing sets via antigen-antibody reactions. Note that the antigen identification is not done by a single or multiple recognizing sets but by antigen-antibody interactions. The more details of mechanism of the immune system are referred to Huang[10,13]. From the above viewpoint, for solving the combinatorial optimization problems, the antibody and antigen can be considered as the solution and objective function, respectively. Thus the concept of immune system can be embodied to construct the so called immune based algorithm for solving the optimization problems.

3.2. *Procedure of Immune Based Algorithm*

The main steps of proposed immune based algorithm (IA) (Hsieh and You[7]) are as follows.

Step 1. (Initial population)
Generate an initial population of strings randomly.

Step 2. (Evaluation)
Evaluate each individual in current population and calculate the corresponding fitness (objective) value for each individual.

Step 3. (Selection)
Select the best s individuals according to fitness values.

Step 4. (Clone)
Clone the best s individuals (antibodies) selected in Step 3. Note that the clone size for each select individual is an increasing function of the affinity with the antigen.

Step 5. (Genetic operation)
The set of the clones in Step 4 will execute the genetic operation process, i.e., crossover and mutation (Michalewicz[14]).

Step 6. (Update memory set)
Calculate the new fitness values of these new individuals (antibodies) from Step 5. Select those individuals who are superior to the individuals in the memory set, and then the superior individuals replace the inferior individuals in the memory set. If the memory set is updated, the individuals will be eliminated while their structures are too similar.

Step 7. (Stopping criterion)
Check the stopping criterion. If not stop then go to Step 2. Otherwise go to next step.

Step 8. (Report solution)
Stop. The optimal or near optimal solution(s) can be obtained from the memory set.

To possess the diversity in the IAs, a suppression process (diversity embodiment) is needed and shown on the Step 6 of IA procedure. In Step 6, the diversity in each pair of antibody i (Ab_i) and antibody j (Ab_j) is computed by their affinity (f_{ij}) as below.

$$f_{ij}=\left\| Ab_i - Ab_j \right\|, \ \forall\, i, j\,.$$

Once the affinity between each pair of antibodies in memory set is obtained, the antibodies will be eliminated if their affinity is less than the predefined threshold. Therefore, the diversity of the antibodies in the memory set can be possessed.

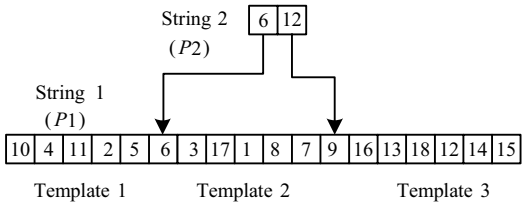

Fig. 2 Two strings for the allocations of 18 labels in 3 templates

3.3. *The Representation Mechanism*

Our implementation is based upon the permutations of $\{1, 2,\ldots, m\}$ and the first n-1 numbers of permutation of $\{1, 2, \ldots, m\text{-}1\}$, where m is the number of types of labels needed and n is the number of templates used. The main steps are shown below.

Step 1: Generate a permutation of $\{1, 2,\ldots, m\}$, say P_1, and generate the first n-1 numbers from a new permutation of $\{1, 2,\ldots, m\text{-}1\}$, say P_2.

Step 2: Divide P_1 into n templates based upon the elements of P_2. (see Example 2 for details)

Step 3: Construct the allocation of templates based upon the division on step 2.

Example 2. Consider the LPP with m=18 and n=3. That is, there are three templates available to produce eighteen distinct types of labels in the LPP. Suppose P_1=10-4-11-2-5-6-3-17-1-8-7-9-16-13-18-12-14-15 be a random permutation of {1, 2, …, 18} and P_2=6-12 be the first 2 numbers randomly generated from a new permutation of {1, 2,…, 17}. Note that 6 and 9 are the 6^{th} and the 12^{th} elements in P_1. Thus, we allocate labels 10, 4, 11, 2, 5, 6 to template 1, labels 3, 17, 1, 8, 7, 9 to template 2, and labels 16, 13, 18, 12, 14, 15 to template 3, respectively. Fig. 2 shows the two strings for the allocations of 18 labels in 3 templates. Once the allocation of labels in each template is obtained, we may decide the number of prints for each template by the procedure in Section 3.4.

3.4. *Number of Prints for LPP*

For each template, we may use the following main results by Yiu et al.[1] to compute the quantities of prints

for each template and then evaluate the objective value based upon objective function (1). Note that once the number of labels is larger than M, i.e., the maximal number of labels for each template, we delete this infeasible solution.

Lemma 3.1. (Yiu et al.[1]) *Assume there are two labels, A and B, assigned to the same template with the order quantity* Q_A and Q_B *such that* $Q_A<Q_B$. *Assume that we have allocated k slots to A. Let the allocation to B be* $kq+r$, *where* $Q_B=qQ_A+r$, $0\le r<Q_A$. Let $W_A= k\lceil Q_A/k\rceil-Q_A$, and $W_B=(kq+r)\lceil Q_A/k\rceil-Q_B$. *If* $W_B<k$, *then the optimal number of prints is* $\lceil Q_A/k\rceil$, *and the optimal allocation to B is given by*

$$r=\left\lceil \frac{Q_B-kq\lceil Q_A/k\rceil}{\lceil Q_A/k\rceil}\right\rceil .$$

Lemma 3.2. (Yiu et al.[1]) *Assume there are two labels, A and B, assigned to the same template with the order quantity* Q_A and Q_B *such that* $Q_A<Q_B$. *Assume that we have allocated k slots to B. Let the allocation to A be r. Let* $W_A=k\lceil Q_B/k\rceil-Q_A$, and $W_B=\lceil Q_B/k\rceil-Q_B$. *If* $W_A<k$, *then the optimal number of prints is* $\lceil Q_B/k\rceil$, *and the optimal allocation to A is given by* $\lceil Q_A/\lceil Q_B/k\rceil\rceil$.

Lemma 3.3. (Yiu et al.[1]) *Assume there are* J_N *labels assigned to the same template and are ranked according to the order quantities in ascending order as* $(Q_1, \ldots, Q_{J-1}, Q_J, Q_{J+1}, \ldots, Q_{JN})$. *Assume that we have allocated k slots to the Jth label. Let the allocation to the other label, denoted by j, is given by* kq_j+r_j, $j\ne J$, *where* $Q_j=q_jQ_J+r_j$, $0\le r_j<Q_J$. *Let* $W_j=(kq_j+r_j)\lceil Q_j/k\rceil/j)-Q_j$ *and*

$$W_{m+1}=\delta\left(M-\sum_{j=1}^{J_N}(kq_j+\gamma_j)\right)\lceil Q_J/k\rceil$$

The minimum number of prints is given by $\lceil Q_J/k\rceil$, *and the corresponding minimum allocation to label j is given by*

$$r_j=\left\lceil \frac{Q_j-kq_j\lceil Q_J/k\rceil}{\lceil Q_J/k\rceil}\right\rceil .$$

These are the optimal allocation for the template if they fulfill the optimality condition:

$$(1-\delta)\sum_{j\ne J}W_j<(1-\delta)k+\delta M\cdot$$

The feasibility condition is:

$$\sum_{j=1}^{J_N}(kq_j+r_j)\le M\cdot$$

4. Numerical Results and Discussions

To test the performance of the applied IA, we solve the two main test problems of LPP by Yiu et al.[1]. In addition, to further evaluate the performance of the applied IA for larger problems, we also test the other four main problems in this paper. These four new test problems are constructed by test problem 1 and test problem 2 by Yiu et al.[1] with the increase of templates and labels. More specifically, the six main problems tested in this paper are (with δ=0.9):

Test problem 1: m=18, M=14 and vary n=3, 4, 5.
Test problem 2: m=22, M=15 and vary n=3, 4, 5.
Test problem 3: m=26, M=16 and vary n=4, 5, 6.
Test problem 4: m=30, M=17 and vary n=4, 5, 6.
Test problem 5: m=34, M=18 and vary n=4, 5, 6.
Test problem 6: m=38, M=19 and vary n=4, 5, 6.

Since we vary n=3, 4, 5 for test problems 1 and 2, and vary n=4, 5, 6 for test problems 3 to 6, the total number of problems is up to 18. The corresponding data of these six main test problems are listed in Table 1. For each of 18 test problems, we set population size=200, maximum iteration=500, affinity=0.1, crossover rate=0.96, mutation rate=0.01, and experiment 20 trials by our IA. All numerical results are computed by Intel-Pentium 4-3.0 GHz PC using algorithms coded in MATLAB 7.1. Numerical results of IA are reported in Table 2, Table 3 and Fig. 3.

From Table 2, Table 3 and Fig. 3, we observe that:

(a) The best solutions by IA are all better than or equal to the best known solutions in the literature. This implies that IA performs better than the other conventional approaches, including SA and heuristics. Moreover, as shown in Table 2, the best solutions of IA for test problem 1 with five templates and test problem 2 with three and four templates have the objective values of 1.3733%, 7.5379% and 3.9691%, respectively, which are superior to 1.5589%, 8.0959% and 4.2348%, respectively, the best well known solutions in the literature.

(b) The applied IA can obtain multiple best solutions for LPP. For example, in test problem 1 of Table 3, there are two best solutions obtained by IA for cases of number of templates 3, 4 and 5, respectively. It implies that the applied IA can effectively obtain multiple best solutions for LPPs.

Table 1. The data of six main test problems for LPP

Label	Problem 1	Problem 2	Problem 3	Problem 4	Problem 5	Problem 6
	Quantity	Quantity	Quantity	Quantity	Quantity	Quantity
1	2200	600	600	600	600	600
2	200	700	700	700	700	700
3	500	2350	2350	2350	2350	2350
4	100	850	850	850	850	850
5	250	625	625	625	625	625
6	550	800	800	800	800	800
7	550	4100	4100	4100	4100	4100
8	550	850	850	850	850	850
9	2500	800	800	800	800	800
10	2450	1025	1025	1025	1025	1025
11	350	4050	4050	4050	4050	4050
12	1150	3300	3300	3300	3300	3300
13	3850	950	950	950	950	950
14	1400	1050	1050	1050	1050	1050
15	2700	5300	5300	5300	5300	5300
16	1400	3750	3750	3750	3750	3750
17	3050	6300	6300	6300	6300	6300
18	5550	6275	6275	6275	6275	6275
19		2275	2275	2275	2275	2275
20		3650	3650	3650	3650	3650
21		2650	2650	2650	2650	2650
22		4850	4850	4850	4850	4850
23			2200	2200	2200	2200
24			200	200	200	200
25			500	500	500	500
26			100	100	100	100
27				250	250	250
28				550	550	550
29				550	550	550
30				550	550	550
31					2500	2500
32					2450	2450
33					350	350
34					1150	1150
35						3850
36						1400
37						2700
38						1400

Note: (1) Problem 1 and Problem 2 are from Yiu et al.[1].

(2) Problem 4 to Problem 6 are constructed by Problem 2 and Problem 1.

Table 2. Numerical results of LPP by various approaches

Test Problem	*M*	No. of labels (*m*)	No. of templates (*n*)	Wastage (%)			
				Factory [(1)]	SA [(1)]	IA [(2)]	CPU [(4)]
1	14	18	3	--- [(3)]	5.8544	5.8544	705.77
			4	5.645	3.2429	3.2429	693.84
			5	---	1.5589	1.3733*	703.72
2	15	22	3	---	8.0959	7.5379*	948.61
			4	---	4.2348	3.9691*	830.95
			5	4.921	2.4682	2.4682	859.79
3	16	26	4	---	---	6.1172	1071.01
			5	---	---	3.5375	1033.92
			6	---	---	2.2574	1031.01
4	17	30	4	---	---	7.5524	1306.76
			5	---	---	3.9489	1235.70
			6	---	---	2.3284	1229.33
5	18	34	4	---	---	8.2784	1612.53
			5	---	---	3.9945	1448.63
			6	---	---	3.0151	1397.81
6	19	38	4	---	---	9.2076	1899.32
			5	---	---	5.1715	1700.00
			6	---	---	3.7438	1591.07

(1) Numerical results reported in Yiu et al.[1].
(2) IA: population=200, maximum generations=500, affinity=0.1, crossover=0.96, mutation=0.01.
(3) ---: not solved.
(4) mean CPU time (second).
* indicates that the solution is superior to the best known solution.

Table 3. Multiple solutions of test problem 1 by IA

Tempaletes (*n*)	Wastage (%)	Label/Templat	Template 1	Template 2	Template 3	Template 4	Template 5
3	5.8544	Label	2-4-5-10-11	1-3-6-7-8-17	9-12-13-14-15-16-18		
		P_{ij}	1-1-1-9-2	4-1-1-1-1-6	2-1-3-1-2-1-4		
		k_i	273	550	1400		
	5.8544	Label	2-3-4-5-10	1-6-7-8-11-17	9-12-13-14-15-16-18		
		P_{ij}	1-2-1-1-9	4-1-1-1-1-6	2-1-3-1-2-1-4		
		k_i	273	550	1400		
4	3.2429	Label	6-9-10-17	1-3-8-11-13	2-4-5-7-12-16	14-15-18	
		P_{ij}	1-4-4-5	4-1-1-1-7	1-1-1-2-4-5	2-4-8	
		k_i	625	550	288	700	
	3.2429	Label	15-16-18	1-6-7-8-13	3-9-10-17	2-4-5-11-12-14	
		P_{ij}	4-2-8	4-1-1-1-7	1-4-4-5	1-1-1-2-4-5	
		k_i	700	550	625	288	
5	1.3734	Label	15-16-18	2-5-17	1-6-7-8-13	3-9-10-14	4-11-12
		P_{ij}	4-2-8	1-1-12	4-1-1-1-7	1-5-5-3	1-3-10
		k_i	700	255	550	500	117
	1.3734	Label	1-6-7-8-13	5-11-14-15	17-18	3-9-10-16	2-4-12
		P_{ij}	4-1-1-1-7	1-1-4-8	5-9	1-5-5-3	2-1-11
		k_i	550	350	617	500	105

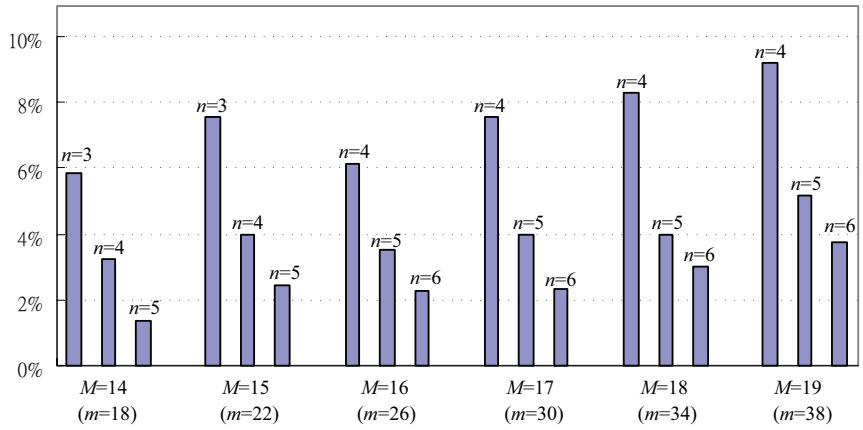

Fig. 3 Numerical results of wastage (%) for six main test problems

(c) Table 2 shows that CPU times for various sizes of test problems are stable. For example, the average CPU time is 1071.01 for case of m=26, n=4, and it is 1031.01 for case of m=26, n=6. While the average CPU time is 1306.76 for case of m=30, n=4, and it is 1229.33 for case of m=30, n=6. As known, there are several factors will affect the CPU time of an algorithm, e.g., the language used in the program, the programmer's skills, and computer types etc. Hence, we have not compared the CPU times of IA with those of the other approaches in this paper. However, Table 2 shows that the proposed approach can be used to solve the LPP within an acceptable time period.

(d) To further evaluate the impact of increasing number of templates for each test problem, based upon 20 trials of the applied IA, we test the following statistical hypothesis:

H_0: $V(n,m)=V(n+1,m)$
H_1: $V(n,m)\neq V(n+1,m)$

where m=18, 22, 26, 30, 34, 38 and n=3, 4, 5, or 6. Test results of p-values show that we all accept H_1. That is, all instances have significant differences in the objective value (wastage) when the number of templates is different. Fig. 3 further illustrates that, when m (the number of labels) is fixed, the wastage will decrease with the increase of n (the number of templates). For example, when m=26, n=4, the objective value (wastage) is 6.1172%, while it reduces to 2.2574% for m=26, n=6.

5. Conclusions

In this paper:

(a) An IA based evolutionary approach has been applied to solve the LPP. Based upon the permutation of {1,2,…, m}, an efficient coding process is proposed to solve LPP.

(b) Numerical results of test problems of LPP have shown the effectiveness of IA. For some test problems, IA can obtain better solutions than those best well known solutions in the literature.

(c) Limited numerical results of test problems have shown that the applied IA can obtain multiple solutions for the LPP.

(d) Numerical results show that the applied IA can solve the LPP within an acceptable time period.

Acknowledgements

The authors would like to thank anonymous reviewers for their helpful comments and suggestions that have greatly improved the presentation of this paper. This research is partially supported by National Science Council, Taiwan, under grant No. NSC 97-2918-I-150-002 and NSC 98-2221-E-150-031.

References

1. K.F.C. Yiu, M.L. Mak and H.Y.K. Lau, A heuristic for the label printing problem, *Comput. Oper. Res.* 34 (2007) 2576-2588.

2. S. Kirkpatrick, C.D. Gelatt Jr and M.P. Vecchi, Optimization by simulated annealing, *Science* 220 (1983) 671-680.
3. E. Sandgren, Nonlinear integer and discrete programming in mechanical design optimization, *Mech. Des.* 112 (1990) 223-229.
4. B. Borchers and J.E. Mitchell, An improved branch-and-bound algorithm for mixed integer nonlinear programming, *Comp. Oper. Res.* 21 (1994) 359-367.
5. R. Fletcher and S. Leyffer, Numerical experience with lower bounds for MIQP branch-and-bound, *SIAM J. Optimiz.* 8 (1998) 604-616.
6. S. Leyffer, Integrating SQP and branch-and-bound for mixed integer nonlinear, *Comput. Optim. Appl.* 18 (2001) 295-309.
7. Y.C. Hsieh and P.S. You, An effective immune based two-phase approach for the optimal reliability-redundancy allocation problem, *Appl. Math. Comput.* 218 (2011) 1297-1307.
8. J.D. Farmer, N.H. Packard and A.S. Perelson The immune system, adaptation, and machine learning, *Physica* 22D (1986) 187–204.
9. L.N. De Castro and J. Timmis, *Artificial immune systems: a new computational intelligence approach*, Springer-Verlag, New York, 2002.
10. S.J. Huang, Enhancement of thermal unit commitment using immune algorithms based optimization approaches, *Electr. Pow. Energ. Syst.* 21 (1999) 245–252.
11. I.L. Weissman and M.D. Cooper, How the immune system develops, *Sci. Am.* 269 (1993) 33-40.
12. N.K. Jerne, The immune system, *Sci. Am*. 229 (1973) 52-60.
13. S.J. Huang, An immune-based optimization method to capacitor placement in a radial distribution system, *IEEE Trans. Power Deliver.* 15 (2000) 744–749.
14. Z. Michalewicz, *Genetic algorithms + data structures = evolution programs*, third Ed., Springer-Verlag, Berlin, 1996.