2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

A Discrete Artificial Bee Colony Algorithm for the Economic Lot Scheduling Problem with Returns

Onder Bulut,
Dept of Engineering,
Yasar University,
İzmir, Turkey,
onder.bulut@yasar.edu.tr

M. Fatih Tasgetiren
Dept of Engineering
Yasar University,
İzmir, Turkey
fatih.tasgetiren@yasar.edu.tr

***Abstract*—In this study, we model the Economic Lot Scheduling problem with returns (ELSPR) under the basic period (BP) policy with power-of-two (PoT) multipliers, and solve it with a discrete artificial bee colony (DABC) algorithm. Tang and Teunter [1] is the first to consider the well-known economic lot scheduling problem (ELSP) with return flows and remanufacturing opportunities. Teunter et al. [2] and Zanoni et al. [3] recently extended this first study by proposing heuristics for the common cycle policy and for a modified basic period policy, respectively. As Zanoni et al. [3], we restrict the study to consider independently managed serviceable inventory to test the performance of the proposed algorithm. Our study, to the best of our knowledge, is the first to solve ELSPR using a meta-heuristic. ABC is a swarm-intelligence-based meta-heuristic inspired by the intelligent foraging behaviors of honeybee swarms. In this study, we implement the ABC algorithm with some modifications to handle the discrete decision variables. In the algorithm, we employ two different constraint handling methods in order to have both feasible and infeasible solutions within the population. Our DABC is also enriched with a variable neighborhood search (VNS) algorithm to further improve the solutions. We test the performance of our algorithm on the two problem instances used in Zanoni et al. [3]. The numerical study depicts that the proposed algorithm performs well under the BP-PoT policy and it has the potential of improving the best known solutions when we relax BP, PoT and independently managed serviceable inventory restrictions in the future.**

I. Introduction

IN the basic ELSP setting, which is firstly described by Rogers [4], a single machine processes items without allowing any shortages where all the processing and demand rates are known constants. Aim is to find the production lot sizes and the order of production that minimizes the sum of average inventory holding and production setup costs. Feasible solutions of this problem satisfy single machine (one-by-one processing), no-stockout and capacity constraints, which guarantees that the required processing and setup times do not exceed the available processing capacity of the machine. It is shown by Hsu [5] that even the feasibility test for ELSP is NP-Complete. In order to generate solutions to this difficult problem, researchers have been working on it under some basic assumptions and different modeling approaches that narrows down the feasible region of the original problem. Vast majority of the related literature assumes at least one of the equal-lot-size and zero-switch restrictions, and models the problem using one of the common cycle, basic period, and extended basic period policies/approaches.

For any given item, zero-switch and equal-lot-size assumptions dictate to trigger a new production cycle if and if the corresponding inventory level hits zero, and to produce in equal lot sizes at each production cycle of that item, respectively. Based on these two assumptions, Hanssmann [6] proposed common cycle (CC) policy that assumes equal cycle lengths for all the items, and later Bomberger [7] extended CC policy to basic period (BP) policy by allowing different cycle times for different items that are integer multiples of a basic period. However, for both CC and BP policies the basic period or the common cycle should be long enough to accommodate the required setup and production times of all the items. To relax this restriction, Elmaghraby [8] formulated ELSP using extended basic period (EBP) policy that allows starting production of different items within different periods.

Bulut and Tasgetiren [9], Chatfield [10], Sun et al. [11, 12] and Khouja [13] are some of the recent seminal studies that model ELSP using the above approaches and propose meta-heuristic optimizers. There are also studies in the literature that consider time-varying lot sizes (e.g. Raza and Akgunduz [14], Dobson [15], Zipkin [16]). For a comparison of different modeling approaches and for a detailed review of the literature we direct the reader to the recent work of Bulut and Tasgetiren [9].

Tang and Teunter [1] is the first to consider ELSP with return flows (ELSPR) by preserving the deterministic nature of the problem. That is, likewise the parameters of the traditional ELSP, the new problem parameters, the return proportions of the items and the remanufacturing rates, are also known constants. The main conceptual difference between the traditional ELSP and ELSPR is the distinction between recoverable and serviceable inventories. Serviceable inventory is common for both ELSP and ELSPR; it is the

stock of all the manufactured or remanufactured items. However, in ELSPR, returns are first stocked in recoverable inventory and are then sent to the serviceable inventory as they are remanufactured.

As both manufacturing and remanufacturing operations replenish the serviceable inventory, the cycle lengths of these operations should be jointly decided for each of the items. This joint management can only be possible if the starting times of all manufacturing and remanufacturing cycles are considered in the modeling phase. It is necessary because the relative timing of manufacturing and remanufacturing operations affects the amount of serviceable inventory. However, considering the starting points of the cycles increases the complexity of the models and eliminates the main common advantage of standard Common Cycle, Basic Period, and Extended Basic Period policies derived for the traditional ELSP. All these approaches enable to model (and solve) the traditional ELSP without explicitly considering the starting points of the cycles.

Tag and Teunter [1] and Teunter et al. [2] modeled the ELSPR under the Common Cycle policy by considering jointly managed serviceable inventory. For the mixed-integer linear program formulation of the problem, Tag and Teunter [1] proposed an exact algorithm that finds out the length of the common cycle and the processing sequence (the starting times of manufacturing and remanufacturing cycles). However, the proposed algorithm is rather complex and time consuming. In order to overcome these disadvantages, Teunter et al. [2] developed a heuristic solution algorithm for the same setting.

Zanoni et al. [3] relaxed the common cycle restriction and introduced a heuristic algorithm that solves the problem under Basic Period policy with Power-of-Two (PoT) frequencies. On the other hand, in order to reduce the model complexity, they restricted the study to consider independently managed serviceable inventory. This means that, they doubled the number of independent items by just splitting the demand of any item into "manufacturing demand" and "remanufacturing demand" using the return proportion for that item. When compared to the joint management of inventory, such an approximation of the objective function increases the holding cost since it ignores coordination opportunities. Teunter et al. [17] also considered ELSPR but they assumed dedicated lines for manufacturing and remanufacturing.

In this study, as Zanoni et al. [3], we assume independently managed serviceable inventory and BP-PoT policy. However, there is a difference between ours and theirs: Although Zanoni et al. [3] formulated the problem under BP policy, if a generated solution violates BP-feasibility they modified the solution by compressing the idle time at the end of the whole cycle that is an integer multiple of a basic period. Therefore, they actually solve the problem under a Modified-BP policy. On the other hand, we totally stick to the (pure) BP policy since our main aim in this study is to apply a meta-heuristic to ELSPR and present the potential of our algorithm for future studies/extensions.

We solve ELSPR for the above described setting with a Discrete Artificial Bee Colony (DABC) algorithm. Our DABC is derived from the well-known Artificial Bee Colony Algorithm (ABC), which is a swarm intelligence based optimizer and originally designed for continuous variables by Karaboga [18-22]. Recently, Bulut and Tasgetiren [9] illustrated that DABC-type algorithms perform well for the traditional ELSP. Our study is the first to consider a DABC algorithm to solve ELSPR. Computational results show that the proposed DABC algorithm is capable of finding the best-known solution for one of the two instances that Zanoni et al. [3] report their Modified-BP-PoT results. We believe that such a performance of our algorithm for independently managed serviceable inventory under the BP-PoT policy, which is more restrictive than the Modified-BP, manifests its potential for the future studies.

Subsequent sections of the paper are organized as follows: Section II and Section III are devoted to the problem formulation and the proposed DABC algorithm, respectively. Section IV presents the computational results and Section V provides concluding remarks and future research directions.

II. Problem Formulation

Economic Lot Scheduling Problem with Returns (ELSPR) under BP-PoT policy and independently managed serviceable inventory can be outlined as follows:

- Demands for the items are experienced at known and constant rates.
- Certain proportions of the manufactured and delivered items are returned back to the facility and remanufactured.
- Manufacturing and remanufacturing operations are held on a single resource (a machine or a production line). That is, two or more items cannot be processed (manufactured or remanufactured) at the same time.
- Manufacturing and remanufacturing rates are known constants.
- Return proportions are used to split the demands into manufacturing and remanufacturing demands in order to independently control the effects of manufacturing and remanufacturing operations on the serviceable inventory.
- For any given item and a processing type (manufacturing or remanufacturing), all processing cycles are in equal length.
- Manufacturing or remanufacturing cycle times of an item are power-of-two (PoT) multiples of the basic period.

In this setting, the aim is to find the length of the basic period and the PoT multipliers that minimize the sum of the average setup and the holding costs. The following notation is used for the mathematical formulation of the problem:

Parameters

$D = Number\ of\ items$

$A_i^m = Setup\ cost\ per\ manufacturing\ lot\ for\ item\ i$

$A_i^r = Setup\ cost\ per\ remanufacturing\ lot\ for\ item\ i$

$s_i^m = Setup\ time\ per\ manufacturing\ lot\ for\ item\ i$

$s_i^r = Setup\ time\ per\ remanufacturing\ lot\ for\ item\ i$

$d_i = Demand\ rate\ for\ item\ i$

$\beta_i = Return\ proportion\ of\ item\ i$

$p_i^m = \ units\ of\ item\ i\ manufactured\ per\ unit\ time$

$p_i^r = units\ of\ item\ i\ remanufactured\ per\ unit\ time$

$h_i^s = holding\ cost\ rate\ for\ servicable\ inventory\ of\ \ i$

h_i^r

$= holding\ cost\ rate\ for\ recoverable\ inventory\ of\ i\ i$

$\rho_i^m = \frac{(1-\beta_i)d_i}{p_i^m},\ \rho_i^r = \frac{\beta_i d_i}{p_i^r}\ (utilization\ factors)$

$H_i^s = (1/2)h_i^s[n_i^m W(1-\beta_i)d_i(1-\rho_i^m) + n_i^r W\beta_i d_i(1-\rho_i^r)]$

$H_i^r = (1/2)h_i^r n_i^r W\beta_i d_i(1-\rho_i^r)$

Decision variables

$W = Length\ of\ a\ basic\ period$

$n_i^m = Multiplier\ for\ manufacturing\ lots\ of\ item\ i$

$n_i^r = Multiplier\ for\ remanufacturing\ lots\ of\ item\ i$

First, BP yields a cyclic schedule which repeats itself in every $max\ (n_i^m, n_i^r)$ periods, which is the length of a complete cycle, because the multipliers are PoT. Second, Under BP policy, if one processes the items using a specific ordering rule (e.g. in the order of increasing multiplier values), then sequence feasibility is satisfied. Thus, it is enough to solve the following non-linear model to characterize the optimal BP policy.

$$\min \sum_{i=1}^{D}\left(\frac{A_i^m}{n_i^m W} + H_i^s\right) + \sum_{i=1}^{D}\left(\frac{A_i^r}{n_i^r W} + H_i^r\right) \quad (1)$$

$s.t$

$$\sum_{i=1}^{D}(s_i^m + s_i^r + n_i^m\rho_i^m W + n_i^r\rho_i^r W) \le W \quad (2)$$

where the (capacity) constraint ensures that all the manufacturing and remanufacturing operations and the corresponding setups can be done within a basic period.

III. Discrete Artificial Bee Colony Algorithm

Inspired by the intelligent foraging behaviors of honeybee swarms, the artificial bee colony (ABC) algorithm was proposed by Karaboga [18-22] who implemented a new swarm intelligence based optimizer. In this algorithm, foraging artificial bees are classified into three groups, namely, *employed bees*, *onlookers*, and *scouts*. *Employed bees* are responsible for food collection from the food source that the bee swarm is exploiting. *Onlookers* wait in the hive and decide on whether a food source is acceptable or not. They make the decision by observing the dancing employed bees' performances. A *scout* randomly searches for new food sources based on internal motivation or some possible external clues. In the ABC algorithm, each solution to the problem under consideration is called a *food source* and represented by a *d*-dimensional real-valued vector where the fitness of the solution corresponds to the *nectar amount* of the associated food resource. As with other intelligent swarm-based approaches, ABC algorithm is an iterative process. The algorithm starts with a population of randomly generated solutions (or food sources) then the following steps are repeated until a termination criterion is met [18-22]:

1. Initialize the foraging process.
2. Send the employed bees to exploit the discovered food sources.
3. Using the onlooker bees, choose the food sources and determine their nectar amounts.
4. Send scouts to search for new food sources.
5. Remember the best food source found so far.
6. If a termination criterion has not been satisfied, go to step 2; otherwise stop the procedure and report the best food source found so far.

The above ABC algorithm, originally designed for the continuous nature of optimization problems, cannot directly be used for discrete/combinatorial cases. In this study, we implement some modifications to the above ABC algorithm to handle discrete decision variables. The details of the modifications are as discussed below.

A. Solution Representation

In our solution representation, the food source is defined as a set of multipliers $n = (n_1^m, n_2^m, .., n_D^m,\ n_1^r, n_2^r, .., n_D^r)$which is given in Figure 1.

	1	*2*	....	*D*
π	n_1^m	n_2^m	....	n_D^m
	n_1^r	n_2^r	....	n_D^r

Fig. 1. Solution representation

The length of the basic period, W is not explicitly included in the solution representation. Instead, parallel with Zanoni et al. [3], for a given *n* vector, it is analytically determined as

$$W^* = max\{W_{max}, W_{min}\} \quad (3)$$

where W_{max} is the unconstrained solution of (1) and W_{min} directly comes from the constraint (2), and calculated by

$$W_{max} = \frac{2\sum_{i=1}^{D}\left(\frac{A_i^m}{n_i^m} + \frac{A_i^r}{n_i^r}\right)}{h_i^s[n_i^m(1-\beta_i)d_i(1-\rho_i^m) + n_i^r\beta_i d_i(1-\rho_i^r)] + h_i^r n_i^r\beta_i d_i(1-\rho_i^r)}$$

$$W_{min} = \frac{\sum_{i=1}^{D}(s_i^m + s_i^r)}{1 - \sum_{i=1}^{D}(n_i^m\rho_i^m + n_i^r\rho_i^r)}$$

B. Feasibility

Any food source represents a feasible solution if the constraint, i.e., the inequality (2), is satisfied. If this is the case, the optimal basic period is given by Eq. (3). On the other hand, if Ineq. (2) does not hold, the current solution, i.e., the current food source, is not feasible. For an infeasible solution, violation is measured by

$$v(\pi) = \sum_{i=1}^{D}(s_i^m + s_i^r + n_i^m\rho_i^m W + n_i^r\rho_i^r W) - W \quad (4)$$

In case of infeasibility, we still employ W^* to compute the objective function value. However, some sophisticated

constraint handling methods are used to handle the infeasibility as explained in the next section.

C. Constraint Handling Methods

In a swarm process, swarm operators may yield infeasible solutions. In this case, care must be taken with them violating the constraints. There exist different approaches to handle the constraints Coello [23]. In this paper, two popular approaches are employed to handle the constraints:

a. Superiority of Feasible Solutions

Deb [24] was the first to propose the superiority of feasible solutions (SF) for constrained optimization based on lexicographic ordering where constraint violation and objective function value are distinguished. The aim is to optimize both constraint violation and objective function by a lexicographic order where constraint violation precedes objective function value. In SF, when two solutions x_a and x_b are evaluated, x_a is deemed to be superior to x_b under the following conditions for a minimization problem:

- x_a is feasible and x_b is not.
- x_a and x_b are both feasible and x_a has a smaller objective function value than x_b.
- x_a and x_b are both infeasible, but x_a has a smaller overall constraint violation $v(x)$ as computed by using Eq. (4).

b. ε -Constraint (EC)

Takahama and Sakai [25-28] developed the ε-constrained method again based on lexicographic ordering with relaxation of the constraints. In the ε -constraint handling method, the relaxation of the constraints is controlled by using the ε parameter. The proper control of the ε parameter is essential to obtain high quality solutions for problems with equality constraints [25-28]. The ε level is updated until the generation counter t reaches the control generation t_C. After the generation counter exceeds t_C, the ε level is set to zero to obtain solutions with no constraint violation. The idea behind EC method is that solutions with the total violation less that $\varepsilon(t)$ are treated as feasible solutions when comparing two solutions. The ε -constraint handling method can be summarized as follows:

$$\varepsilon(0) = v(\pi_\theta) \tag{5}$$

$$\varepsilon(t) = \begin{cases} \varepsilon(0)\left(1 - \dfrac{t}{t_C}\right)^{cp}, & 0 < t < t_C \\ 0, & t \geq t_C \end{cases}. \tag{6}$$

where π_θ is the top θ-th individual and $\theta = 0.20 \times NP$. The recommended parameter ranges are $T_C = (0.1 \times T_{\max}, 0.8 \times T_{\max})$ and $cp = (2,10)$.

It is important to note that we employed these two constraint handling methods in the proposed GA as follows: Before making the comparison of two individuals, each individual is treated to either feasible or infeasible according to the ε level. Then, SF is used when making comparison.

D. Initial Population

The initial population of NP individuals is constructed randomly. For each food source, n_i^m and n_i^r are randomly generated from $\{1,2,4,..,n_{\max}\}$ where the maximum multiplier was assumed to be $n_{\max} = 16$. To simplify, PoT multipliers are generated by $2^{rand()\%n_{\max}}$.

E. Employed Bee Phase

According to the basic ABC algorithm, the employed bees generate food sources in the neighborhood of their current positions. The DABC algorithm employs a very simple neighboring food source by "*Multiply By 2*" , "*Divide By 2*" and "*Union of N_1 and N_2*". These three strategies are as follows:

$N_1(\pi) = Divide \quad By \quad 2$

$N_2(\pi) = Multiply \quad By \quad 2$

$N_3(\pi) = Union \quad of \quad N_1 \quad and \quad N_2$

Note that these neighboring strategies obey the PoT rule. In other words, dividing by 2 or multiply by 2 does not generate non-PoT values. However, in the case of dividing 1 by 2, the dimension value of the food source is assigned to 2. If the resulting dimension value is greater than $2^{n_{\max}}$, it is randomly assigned to $2^{rand()\%n_{\max}}$. For a 5-item problem, Figure 2-4 illustrates the neighborhoods used in the DABC algorithm.

	1	2	3	4	5
π	1	**4**	2	8	2
	1	**2**	4	4	8
$N_1(\pi)$	1	**2**	2	8	2
	1	**1**	4	4	8

Fig. 2. Divide by 2

	1	2	3	4	5
π	1	**4**	2	8	2
	1	**2**	4	4	8
$N_2(\pi)$	1	**8**	2	8	2
	1	**4**	4	4	8

Fig. 3. Multiply by 2

	1	2	3	4	5
π	1	**4**	2	8	**2**
	1	**2**	4	4	**8**
$N_3(\pi)$	1	**8**	2	8	**1**
	1	**4**	4	4	**4**

Fig. 4. Union

Each method for the generation of neighboring food sources may have different performances during the evolution process. Therefore, each food source in the population is assigned to one of the three strategies to generate a neighboring food source. After generating a neighboring food source by one of three strategies, a local search based on variable neighborhood search (VNS) [29] is applied to further improve the solution quality (nectar amount). As for the selection, a new good source is always accepted if it is better than the current food source (according to SF and ε-constraint level), which is similar to the basic ABC algorithm carrying out a greedy selection procedure. The number of the employed bees is set to the population size NP.

F. Onlooker Bee Phase

In the basic ABC algorithm, an onlooker bee selects a food source π_k depending on its winning probability value, which is similar to wheel selection in GAs [18-22]. However, the tournament selection is widely used in GA applications due to its simplicity and ability to escape from local optima. For this reason, we propose a tournament selection with a size of 2 in the DABC algorithm. In the tournament selection, an onlooker bee selects a food source π_k in such a way that two food sources are randomly picked up from the population and compared to each other, allowing the better one to be chosen. In addition, an onlooker bee utilizes the same strategy that is used by the employed bee to produce a new neighboring solution. Then, a well-devised VNS local search is employed to further improve the nectar amount of the onlooker bee. If the new food source obtained is better than the current one, the new food source will replace the current one and become a new member in the population (according to SF and ε -constraint level). The number of onlooker bees considered is NP.

G. Scout Bee Phase

In the basic ABC algorithm, a scout bee produces a food source randomly in the predefined search space. This will decrease the search efficacy because the best food source in the population often carries better information than others during the evolution process, and the search space around it could be the most promising region. Therefore, in the DABC algorithm, a tournament selection with the size of 2 is again used to discard the worse of two randomly selected food sources that have been picked out from the population. Then, the scout generates a food source randomly as explained before. This food source will be replaced by the food source determined by tournament selection, thus proposing a kind of immigration to the DABC algorithm. There are $0.1\times NP$ scout bees in this phase.

H. Variable Neighborhood Search

In order to intensify the search on the local minima and improve the solution quality of DABC, a common approach is to hybridize it with some local search methods. For this reason, the local search algorithm is, simply VNS), fused into the DABC algorithm. Note that VNS is applied to each food source generated either by the employed bee or onlooker bee. VNS is a meta-heuristic proposed in [29] systematically exploiting the idea of neighborhood change. We employed the following VNS local search in Figure 5. The VNS local search has three neighborhood structures as used in *Employed Bee and Onlooker Bee* phases. In other words, the following neighborhood structures are employed:

$N_1(\pi) = Divide \quad By \quad 2$

$N_2(\pi) = Multiply \quad By \quad 2$

$N_3(\pi) = Union \quad of \quad N_1 \quad and \quad N_2$

The third neighborhood is a combination of N_1 and N_2. In other words, once a dimension value of the food source is randomly changed by N_1, another dimension is also randomly changed by N_2. The size of the each neighborhood is the number of dimensions/items. In other words, for example, $Divide \quad By \quad 2$ is applied 10 times and the best out of 10 is retained.

$\text{Pr}\,ocedure \quad VNS(\pi)$
$k_{\max} := 3$
$k := 1$
$do\{$
  $\pi_1 = N_k(\pi)$
  $if \quad f(\pi_1) < f(\pi) \quad then$
  $\quad\quad with \quad SF \quad and \quad \varepsilon-level$
    $\pi = \pi_1$
    $k := 1$
  $else$
    $k := k + 1$
  $\}while(k \le k_{\max})$
$return \quad \pi$
$endprocedure$

Fig. 5. The VNS Algorithm.

IV. Computational Results

The proposed DABC algorithm was coded in C++ and run on an Intel P4 3.00 GHz PC with 500MB memory. The population size was fixed at 100 and the algorithm was run for 1000 generations with the following constraint handling parameters: $\theta = 0.2\times NP$, $t_C = 0.4 * MaxGen$ and $cp = 2$.

Teunter et al. [2] provided 120 benchmark problem instances. We first test the performance of our algorithm on the Instance#61 and Instance#101for which Zanoni et al. [3] reported their Modified-BP-PoT results. Five runs carried out for both of the instances and the computation time for a run is roughly less than 5 s per 1000 generations. In Table II, for both of the instances, we present the best out our five replications and compare them to the results of Zanoni et al. [3] and Teunter et al. [2].

As seen from Table II, even though Zanoni et al. [3] solved the problem under a Modified-BP policy, which has a larger feasible region than the BP-Policy that we employ, and Teunter et al. [2] coordinated the manufacturing and remanufacturing operations (that means they do not have the

independently managed serviceable inventory assumption) within CC policy, our algorithm was able to find the best known solution for Instance #61. This result reveals the potential of our algorithm for future extensions of ELSPR. Table II also illustrates the fact that considering the joint management of serviceable inventory, i.e., considering the starting times of all manufacturing and remanufacturing cycles as decision variables, significantly improves the solution for Instance #101. This is an expected result because utilization factor for Instance #101, which is 0.95, is much greater than the utilization factor of Instance #61, which is 0.75. It is well known that ELSP-type problems get harder as the utilization factor increases.

TABLE II.
RESULTS FOR INSTANCES 61 AND 101

Instance	Teunter et al. [2]	Zanoni et al. [3]	DABC
61	28.32	**19.21**	**19.21**
101	**51.22**	30.77	56.40

The solution vectors of our DABC algorithm for instances 61 and 101 are presented in Table III where $n^m = (n_1^m, n_2^m, .., n_D^m)$ is the vector of PoT multipliers for the "manufacturing demands", and $n^r = (n_1^r, n_2^r, .., n_D^r)$ is the vector of PoT multipliers for the "remanufacturing demands".

TABLE III.
DABC SOLUTIONS FOR INSTANCES 61 AND 101

Instance	W	n^m	n^r
61	98.02	(1,1,1,1,1,1,1,1,1,1)	(1,1,1,1,1,1,1,1,1,1)
101	74.24	(4,1,1,1,1,8,16,2,8,2)	(1,1,1,1,1,8,8,1,2,1)

TABLE IV.
RESULTS FOR INSTANCES 51 - 60

Instance	Teunter et al. [2]	DABC
51	59.53	**37.85**
52	17.71	**17.29**
53	25.83	**24.95**
54	25.81	**24.02**
55	19.84	**17.03**
56	27.57	**26.84**
57	**23.13**	26.95
58	25.65	**23.16**
59	**27.51**	28.67
60	22.43	**20.18**

We also run our algorithm for the ten instances in the range Instance#51 - Instance#60 and present the results in Table IV. Since the results of Zanoni et al. [3] are only available for the instances 61 and 101, in Table IV, we only compare our results to the results of Teunter et al. [2]. Our algorithm was able to improve the results for eight of the ten instances. Parallel to the discussion on Table II, it can be also concluded form Table IV that joint management of serviceable inventory provides better results for the higher utilization problems.

V. CONCLUSIONS

In this study we present a discrete artificial bee colony (DABC) algorithm to solve the economic lot scheduling problem with returns (ELSPR) under basic period (BP) policy with power-of-two (PoT) multipliers. We also assume independently managed serviceable inventory as Zanoni et al. [3]. Computational results show that the proposed DABC algorithm is capable of finding the best-known solution for one of the two instances that Zanoni et al. [3] reported their Modified-BP-PoT results. It is also shown that our algorithm based on the Basic Period approach succeeds to improve the Common Cycle results especially for the low utilization problems. We believe that such a performance of our algorithm for independently managed serviceable inventory under the BP-PoT policy, which is more restrictive than the Modified-BP, manifests its potential for the future studies.

As a future research direction, we plan to relax all the BP, PoT and independently managed serviceable inventory restrictions and enhance our algorithm accordingly.

REFERENCES

[1] O. Tang and R. H. Teunter, "Economic lot scheduling problem with returns", Production and Operations Management, vol. 15, pp. 488-497, 2006.

[2] R. H. Teunter, O. Tang and K. Kaparis, "Heuristics for economic lot scheduling problem with returns", International Journal of Production Economics, vol. 118, pp. 323-330, 2009.

[3] S. Zanoni, A. Segerstedt, O. Tang, and L. Mazzoldi, "Multi-product economic lot scheduling problem with manufacturing and remanufacturing using a basic period policy", Computers and Industrial Engineering, vol. 62, pp. 1025-1033, 2012.

[4] J. Rogers, "A computational approach to the economic lot scheduling problem", Management Science , Vol. 4, pp.264-291, 1958.

[5] W. Hsu, "On the general feasibility test of scheduling lot sizes for several products on one machine," Management Science, vol. 29, pp. 93-105, 1983.

[6] F. Hanssmann, Operation Research in Production and Inventory, Wiley, New York, 1962.

[7] E.E. Bomberger, "A dynamic programming approach to a lot size scheduling problem," Management Science, vol. 12, pp. 778-784, 1966.

[8] S.E. Elmaghraby, "The economic lot scheduling problem (ELSP): review and extensions," Management Science, vol. 24, pp. 587-598, 1978.

[9] O. Bulut and F. Tasgetiren, "An artificial bee colony algorithm for the economic lot scheduling problem", International Journal of Production Research, DOI: 10.1080/00207543.2013.845315, 2013.

[10] D. Chatfield, "The economic lot scheduling problem: a pure genetic search approach," Computers and Operations Research, vol. 34, pp. 2865-2881, 1987.

[11] H. Sun, H.Huang and W. Jaruphongsa, "The economic lot scheduling problem under extended basic period and power-of-two policy", Optimization Letters, vol. 4, pp. 157-172, 2010.

[12] H. Sun, H-C. Huang, W. Jaruphongsa, "A genetic algorithm for the economic lot scheduling problem under the extended basic period and power-of-two policy", CIRP Journal of Manufacturing Science and Technology, 2(2009) 29-34.

[13] M. Khouja , Z. Michalewicz and M. Wilmot, "The use of genetic algorithms to solve the economic lot size scheduling problem", European Journal of Operational Research, vol. 110, pp. 509-524, 1998.

[14] Raza, A. S., and Akgunduz, A. "A comparative study of heuristic algorithms on economic lot scheduling problem", Computers and Industrial Engineering, vol. 55(1), pp. 94–109, 2008.

[15] G. Dobson, "The ELSP: achieving feasibility using time-varying lot sizes," Operations Research, vol. 35, pp. 764–71, 1987.

[16] P. Zipkin, "Computing optimal lot sizes in the ELSP," Operations Research, vol. 39, pp. 56–63, 1991.

[17] R. H. Teunter, K. Kaparis and O. Tang, "Multi-product economic lot scheduling problem with separate production lines for manufacturing

and remanufacturin ", European Journal of Operational Research, vol. 191, pp. 1241-1253, 2008.

[18] D. Karaboga, An idea based on honey bee swarm for numerical optimization, Technical Report TR06, Computer Engineering Department, Erciyes University, Turkey, 2005.

[19] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, Journal of Global Optimization 39 (2007) 459-471.

[20] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Applied Soft Computing 8, pp. 687-697, 2008.

[21] D. Karaboga, A new design method based on artificial bee colony algorithm for digital IIR filters, Journal of The Franklin Institute, vol. 346, pp. 328-348, 2009.

[22] D. Karaboga, B. Akay, A comparative study of artificial bee colony algorithm, Applied Mathematics and Computation, doi:10.1016/j.amc.2009.03.90, 2009.

[23] C.A. Coello, Theoretical and Numerical Constraint-Handling Techniques Used with Evolutionary Algorithms: A Survey of the State of the Art, Comput. Methods Appl. Mech. Engrg. vol. 191, pp. 1245-1287, 2002.

[24] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering,* vol. 186, pp. 311-338, 2000.

[25] T. Takahama and S. Sakai, "Tuning fuzzy control rules by the α constrained method which solves constrained nonlinear optimization problems," Electronics and Communications in Japan, Part3: Fundamental Electronic Science, vol. 83, no. 9, pp. 1–12, Sept. 2000.

[26] T. Takahama and S. Sakai, "Constrained optimization by ε constrained particle swarm optimizer with ε-level control," in Proc. of the 4th IEEE International Workshop on Soft Computing as Transdisciplinary Science and Technology (WSTST'05), May 2005, pp. 1019–1029.

[27] T. Takahama and S. Sakai, "Efficient Constrained Optimization by the ε Constrained Adaptive Differential Evolution", WCCI 2010 IEEE World Congress on Computational Intelligence July, 18-23, 2010, Barcelona, Spain.

[28] T. Takahama and S. Sakai, "Constrained Optimization by the Constrained Differential Evolution with Gradient-Based Mutation and Feasible Elites," in *IEEE Congress on Evolutionary Computation* Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada, 2006, pp. 1-8.

[29] N. Mladenovic, P. Hansen, Variable neighborhood search, *Computers and Operations Research* 24 (1997) 1097-1100.