A Hybrid Discrete Differential Evolution Algorithm for Economic Lot Scheduling Problem with Time Variant Lot Sizing

Srinjoy Ganguly[1], Arkabandhu Chowdhury[1], Swahum Mukherjee[1], P.N. Suganthan[2], Swagatam Das[3], and Tay Jin Chua[4]

[1] Dept. of Electronic & Telecommunication Engineering, Jadavpur University, Kolkata, India
[2] School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798
epnsugan@ntu.edu.sg
[3] Electronics and Communication Sciences Unit, Indian Statistical Institute, Kolkata, India
swagatam.das@isical.ac.in
[4] Singapore Institute of Manufacturing Technology (SIMTech), 71 Nanyang Drive, Singapore 638075
tjchua@SIMTech.a-star.edu.sg

Abstract. This article presents an efficient Hybrid Discrete Differential Evolution (HDDE) model to solve the Economic Lot Scheduling Problem (ELSP) using a time variant lot sizing approach. This proposed method introduces a novel Greedy Reordering Local Search (GRLS) operator as well as a novel Discrete DE scheme for solving the problem. The economic lot-scheduling problem (ELSP) is an important production scheduling problem that has been intensively studied. In this problem, several products compete for the use of a single machine, which is very similar to the real-life industrial scenario, in particular in the field of remanufacturing. The experimental results indicate that the proposed algorithm outperforms several previously used heuristic algorithms under the time-varying lot sizing approach.

Keywords: Lot scheduling, time-varying lot-sizes approach, discrete differential evolution, cyclic crossover, simple inversion mutation, greedy reordering local search, remanufacturing.

1 Introduction

It is a common practice in industries to produce several products on a single machine due to economic considerations. Typically, these facilities may produce only a single product at a time and have to be set-up (stopped and prepared) at the cost of time and money, before the start of the production run of a new product. A production scheduling problem arises due to the need to co-ordinate the set-up and production of a large number of different items. The main aim of the Economic Lot Scheduling Problem (ELSP) [1] is to find the best lot sizes and production schedule that does not allow any shortages for the items to be produced in the above described environment. Typical examples of such problems are:

V. Snasel et al. (Eds.): SOCO Models in Industrial & Environmental Appl., AISC 188, pp. 1–12.
springerlink.com

- Metal forming and plastics production lines (press lines, and plastic and metal extrusion machines), where each product requires a different die that needs to be set up on the concerned machinery.
- Assembly lines which produce several products and different product models (electric goods, vehicles, etc.).
- Blending and mixing facilities (for paints, beverages, etc.), in which different products are poured into different containers for processing.

Typically, in an industrial scenario, a single machine of very high efficiency is purchased instead of several machines of lesser efficiency. This situation leads to the question of how one should schedule production on this high speed machine. The issue is one of selecting both a sequence, in which the products will be manufactured, and a batch size for each item run. The issue of batching arises because the system usually incurs a set-up cost and/or a set-up time when the machine switches from one product to a different product. Set-up times imply an idle-time during which the machine does nothing, which, in turn, implies a need to carry a large scale production facility. This problem has attracted the attention of many researchers over 40 years, partly because it is a representation of many frequently encountered scheduling problems, and simply because it appears to be unconquerable. In fact, many researchers also take into consideration another interesting variant of the problem wherein they allow the phenomenon of re-manufacture [2] to occur, i.e. items that have been returned by the consumers are re-manufactured and made as fresh as new. Ouyang and Zhu extended the classical ELSP to schedule the manufacturing and remanufacturing on the same single product line [19]. They assumed that the demand rate and return rate are constant and the product line has limited capacity of manufacturing and remanufacturing.

Typically, we may assume that the demand rates are known before-hand and are product-dependent while the set-up cost and set-up time are product-dependent but sequence-independent. Also, the majority of the research in ELSP literature focuses on cyclic production schedules, i.e. the schedule is repeated periodically. As the ELSP is an NP-hard problem, many heuristics have been devised that may solve this problem to near optimality. The three types of approaches generally taken are:

I. **Common cycle approach:** This restricts all the products' cycle times to equal time (an item's cycle time is the duration between the starts of two consecutive runs of that item). This approach has the advantage of always generating a feasible schedule despite the use of a very simple procedure. This procedure, however, gives solutions far from the lower bound in some cases [3].

II. **Basic period approach:** This allows different cycle times for different products, but restricts each product's cycle time to be an integer multiple k of a time period called a basic period. This approach, in general, gives better solutions than the common cycle approach. However, its main drawback is the difficulty of ensuring that the production sequence is feasible [4].

III. **Time-varying lot size approach:** This allows different lot sizes for any given product during a cyclic schedule. It explicitly handles the difficulties caused by set-up times and always gives a feasible schedule as proved by Dobson [5]. It has been found to give fitter solutions in comparison to the previous two approaches.

The research on ELSP under the different policies discussed above mainly comprises of different algorithmic solutions since the restricted versions of the problem are also very difficult. Till date, most researchers have relied on genetic algorithms to solve the problem. There are only two studies in the literature that consider exact algorithms: Grznar and Riggle [6] for BP policy, and Sun *et al*. [7] for EBP policy. However, the exact algorithms are not very time-efficient especially when the utilization factor is high. The purpose of the current research is to develop a Hybrid Discrete Differential Evolution (HDDE) algorithm to solve the ELSP. Our HDDE is based on the time-varying lot sizes approach. In this paper, we present the ELSP formulation and proposed HDDE in Section 2 and the results and discussions in Section 3.

2 ELSP Problem Formulation and Algorithm

The following assumptions are normally used in the formulation of the ELSP:

- Several items compete for the use of a single production facility.
- Demand-rates, production-rates, set-up times and set-up costs are known before-hand and are constant.
- Backorders are not allowed.
- Inventory costs are directly proportional to inventory levels.

The following notations have been adopted:

- Item Index: $i = 1,2,...,M$
- Position Index: $j = 1,2,...,N$
- Constant Production Rate(Units per day): p_i ; $i = 1,2,...,M$
- Constant Demand Rate(Units per day): d_i ; $i = 1,2,...,M$
- Inventory Holding Cost(Price per unit per day): h_i ; $i = 1,2,...,M$
- Set-up Cost(Currency): a_i ; $i = 1,2,...,M$
- Set-up Time(days): s_i ; $i = 1,2,...,M$
- Item produced at position j: I^j ; $j = 1,2,...,N$
- Production-time for item produced at position j: t^j ; $j = 1,2,...,N$
- Idle-time for the item produced at position j: x^j ; $j = 1,2,...,N$
- Cycle Length(days): T

The ELSP in a nutshell is the situation where there is a single facility on which different products have to be produced. We try to find a cycle length and a production sequence $\boldsymbol{I}=(I^1, I^2,..., I^N)$ where $I^j \in (1,2,...,M)$, production time durations $\boldsymbol{t}=(t^1, t^2,..., t^N)$ and idle-times $\boldsymbol{u}=(u^1, u^2,..., u^N)$ so that the given production cycle can be finished within the given cycle. This cycle has to be repeated

over and over again, while at the same time, inventory and set-up costs have to be minimized. We may define μ as:

$$\mu = 1 - \sum_{i=1}^{m} \frac{d_i}{p_i}$$

We must note that μ represents the long-run proportion of time available for set-ups. For infinite horizon problems, $\mu > 0$ is absolutely necessary for the existence of a feasible solution. It can be shown that any production sequence can be converted into a feasible one if $\mu > 0$.

Let F represent the set of all the feasible finite sequences of the products and J_i denote the positions in the schedule where the product having the index i is produced. Let Y_k denote all the jobs in a given sequence starting from k up till that position in the given sequence where the item I^k is produced again. Then, the complete formulation of the ELSP is:

$$\inf_{j \in F} \min_{t \geq 0; x \geq 0; T \geq 0;} \frac{1}{T} \left(\sum_{j=1}^{N} \frac{1}{2} h^j (p^j - d^j) \left(\frac{p^j}{d^j} \right) (t^j)^2 + \sum_{j=1}^{N} a^j \right) \tag{1}$$

Subject to the following boundary conditions:

- $$\sum_{j \in J^i} t^j p_j = d_i T; i = 1,2,..., M \tag{2}$$

- $$\sum_{j \in Y_k} t^j + s^j + x^j = \frac{p^k}{d^k} t^k; k = 1,2,..., N \tag{3}$$

- $$\sum_{j=1}^{N} (t^j + s^j + x^j) = T \tag{4}$$

The condition (2) ensures that enough space is allocated to each product so that it may meet its own demand during one complete cycle. Condition (3) ensures that we produce that much quantity of each product so that its demand is met till the time it is produced again. Condition(4) ensures that the total time taken for the complete cycle is numerically equal to the sum of the production time, setup time and idle time of the various items.

2.1 The Proposed HDDE Algorithm

The HDDE may be categorized into the following steps:

Step 1 The production frequencies are obtained by solving the lower bound (LB) model [8] as stated below. This lower bound is tighter than that obtained by using the so-called independent solution in which each product is taken in isolation by calculating its economic production quantity. The constraint here is that enough time must be available for set-ups. As stated previously, μ represents the average

proportion of time available for set-up. T_i refers to the cycle length for the product i. However, the synchronization constant that states that no two items may be produced simultaneously is ignored. Hence, this results in a lower total daily cost for the ELSP. This scheme was initially proposed by Bomberger [9].

LB model:

$$\min_{T_1,\ldots,Tm} \sum_{i=1}^{M} \left(\frac{a_i}{T_i}\right) + \left(\frac{h_i d_i T_i}{2}\right)\left(1 - \frac{d_i}{p_i}\right)$$

Given that, $\sum_{i=1}^{m} \frac{s_i}{T_i} \le \mu$; $\quad T_i \ge 0;\ i = 1,2,\ldots,M$ (5)

The objective function and the convex set in the above constraint model are convex in $T_i's$. Therefore, the optimal points of the LB model are those points that satisfy the KKT conditions as follows:

$$\sqrt{\frac{a_i + Rs_i}{H_i}} = T_i \text{ for all } i \quad . \tag{6}$$

$R \ge 0$ with complementary slackness $\sum_{i=1}^{m} \frac{s_i}{T_i} \le \mu$, where $H_i = \frac{1 - \frac{d_i}{p_i}}{2}$. The procedure that is adopted to find the optimal T_i's is described below.

Algorithm for the Lower Bound

1. The condition *R*=0 is checked for an optimal solution. The T_i's are found as per the formula $\sqrt{\frac{a_i}{H_i}}$ for all *i*. If $\sum_{i=1}^{m} \frac{s_i}{T_i} < \mu$, then the T_i's are proven to be an optimal solution. Else, the algorithm explained in step 2 is adopted.
2. *R* is taken at an arbitrary value greater than 0. The T_i's are computed as per $\sqrt{\frac{a_i + Rs_i}{H_i}} = T_i$. If $\sum_{i=1}^{m} \frac{s_i}{T_i} < \mu$, *R* is reduced and the afore-mentioned process is repeated. If $\sum_{i=1}^{m} \frac{s_i}{T_i} > \mu$, *R* is increased and the afore-mentioned process is repeated. If $\sum_{i=1}^{m} \frac{s_i}{T_i} = \mu$, the iterations stop and the set of T_i's obtained are optimal.

If the optimal cycle length for an item is $T_i^{'}$, then the production frequency (f_i) of that item may be obtained as per the following formula: $f_i = \frac{\max(T_i^{'})}{T_i^{'}}$.

Step 2 The production frequencies obtained in Step 1, are rounded off to the nearest integers.

Step 3 Using the frequencies obtained in the previous step, an efficient production sequence is obtained using the DE algorithm. This algorithm is discussed later in details.

Step 4 If we assume the approximation that there is no idle time (which works well for highly loaded facilities), we can easily solve for the set of production times using equation (3). This method is referred to as the Quick and Dirty heuristic [10]. Otherwise, we may adopt Zipkin's [11] parametric algorithm.

2.2 Discrete Differential Evolution

The key components of the proposed DDE are:

1) Representation of the Schedule (Chromosome): The proper representation of a solution plays an important role in any evolutionary algorithm. In our paper, a string of positive integers (chromosome) is used to represent a solution. The length of the chromosome is the sum of the production frequencies obtained at the end of step 2. As standard operations such as cross-over and mutation are difficult using such a representation, we use an additional chromosome that possesses the absolute locations of each of the genes in the afore-mentioned chromosome. Suppose the problem at hand is a 4-product ELSP and it is observed at the end of step 2 that their frequencies are 1,2,2,1 respectively. We may represent this situation using two chromosomes A and B. The representation is as follows:

Chromosome A ⟶ (1 2 2 3 3 4)
Chromosome B ⟶ (1 2 3 4 5 6)

Here, Chromosome A represents the item numbers simply, while Chromosome B represents their respective locations. All the operations shall be performed using chromosomes of the form of Chromosome B.

2) The objective and fitness function. The fitness value for any chromosome may be computed as the inverse of Equation (1), which serves as the fitness function. Our objective is to minimize this function (Eq. 1) and therefore maximize the fitness, i.e. our aim is to find the string with the maximum fitness. For any chromosome (representing the production schedule) the production times of the different items may be computed as per step 4 of the main algorithm. After this, we obtain the value of the fitness function using Equation No. (1). The fitness function is represented by $f_{elsp}(V)$ for chromosome V.

3) The Cross-Over operator. The Cyclic Crossover Operator [12] is of immense importance in the proposed HDDE because this operation is carried out at several points in the algorithm. This operator is unique one since it preserves characteristics of both the parents in the offspring. The crossover mechanism may be envisaged via the following example: Let us consider two flow sequences A and B, where A= (1 3 5 6 4 2) and B= (5 6 1 2 3 4). Let the first offspring begin with 1(the starting operation of parent A). Selection of '1' from A implies that '5' should be selected from B, because we want each operation to be derived from one of the two parents. Hence, C= 1 _ 5 _ _ _. This process continues on till after the selection and subsequent insertion of an operation from one of the two parents, the operation in the

corresponding position in the other parent is already present in the offspring. After this, the remaining operations are filled in, as per their orders respective to one another in the other string. As can be seen in case of C, insertion of '5' implies that '1' should be inserted in the list, but as '1' is already present in the list (i.e. the starting operation), the cycle stops (hence, the name Cyclic Crossover) and the remaining operations are filled in from B. Hence, C = (1 6 5 2 3 4). Similarly, considering '5' as the starting operation, another offspring D can be obtained in a similar fashion. Hence, D= (5 3 1 6 4 2) .As our algorithm imposes a restriction that the result of each crossover operation results in only a single offspring, the fitter progeny is selected.

4) Perturbation Operator. In our paper, we have adopted the Simple Inversion Mutation [13] as our perturbation operator. Two random cut-points are selected in the chromosome and that part of the chromosome between these two cut-points is inverted and placed in the original chromosome. For example, let us consider the chromosome represented by **(ABDEGCFJIH).** Suppose the first cut-point is between D and E and the second cut-point is between F and J.

Then the part of the chromosome between the two cut-points is reversed and then placed in the original chromosome to obtain **(ABDFCGEJIH).**

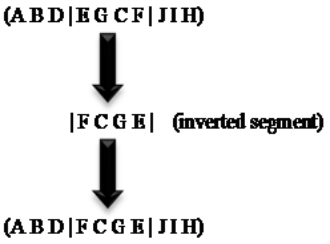

Fig. 1. Simple Inversion Mutation

DE [14] is one of the recent evolutionary optimization algorithms. Like other algorithms of the same class, DE is population-based and serves as a stochastic global optimizer. In the classic DE, candidate solutions are represented by a vector of floating point solutions. The mutation process entails the addition of the weighted difference of two members of the population to another third member of the same population to generate a mutated solution. Then, a cross-over operation occurs so as to generate a trial solution that is in turn compared with the target solution using a selection operator, which determines which of the two shall survive for the next generation. It is obvious that traditional DE equations shall not hold in this regard as they had been developed for continuous numerical optimization. As per the novel DDE algorithm proposed by Pan et. al. [15], the target individual is represented by a chromosome of type B (permutation of numbers). The mutant individual is obtained by perturbing the previous generation best solution in the target population. Hence, we achieve the difference variations by applying a perturbation (mutation) operator on the best solution present in the previous generation of the target population. These

perturbations are achieved by applying the Simple Inversion Mutation operator on the best solution of the previous generation. Since these perturbations are stochastically different, each result is expected to be distinct from the other. To obtain the mutant, the following equation may be used:

$$V_i^t = \begin{Bmatrix} F(X_g^{t-1})..if..r \le mutation_{prob} \\ X_g^{t-1}..otherwise \end{Bmatrix} \quad (7)$$

where, X_g^{t-1} represents the best solution of the previous generation of the target population. $F()$ denotes the perturbation operator(Simple Inversion Mutation). Hence, a random number $r \in (0,1)$ is generated and if it is less than $mutation_{prob}$, then the trial solution is obtained by applying the perturbation operator on X_g^{t-1} else, the mutant is set to X_g^{t-1}. After this, we enter the recombination (cross-over) phase. In this, $V^t{}_i$ is crossed over with X_i^t to obtain the trial solution U_i^t if $v \in (0,1)$ [generated randomly] is greater than $crossover_{prob}$. Else, U_i^t is taken to be equal to V_i^t . The pseudo-code representation for the afore-said is:

$$U_i^t = \begin{Bmatrix} CR(X_i^{t-1}, V_i^t..if..v > crossover_{prob} \\ V_i^t..otherwise \end{Bmatrix} \quad (8)$$

Hence, if U_i^t is fitter than X_i^{t-1} , then $X_i^t = U_i^t$. Else, $X_i^t = X_i^{t-1}$. Over here, $CR()$ refers to the cross-over operator. As discussed earlier, the Cyclic Cross-Over operator has been used in the HDDE. As it is pretty ostensible, our basic aim is to take advantage of the best solution obtained from the previous generation during the entire search process. Unlike its continuous counterpart, the differential evolution is achieved by the stochastic re-ordering of the previous generation's best fit solution in the target population.

Greedy Reordering Local Search (GRLS). In this paper, we propose a novel local search operator which restructures a chromosome iteratively on a probabilistic basis to search for fitter solutions. This re-ordering process goes on till the solutions which are obtained from reordering is better in comparison to the parent solution. The GRLS algorithm for a chromosome p is given in Fig. 2.

3 Results and Discussion

In our tests on ELSP, the proposed HDDE algorithm is coded in MATLAB and the experiments are executed on a Pentium P-IV 3.0 GHz PC with 512MB memory. We

have compared the results that our algorithm provided when tested on Mallya's 5-item problem [17] and the famous Bomberger's 10-item problem (for the μ=0.01 case) [8] with the hybrid GA proposed by Moon et. al.[10] , Khouza's GA [17], Dobson's heuristic [18] and the Common Cycle Solution [3] approach. In each of the runs, the number of iterations (generations) have been taken to be 80 and the population size has been taken as 100, the value of $mutation_{prob}$ has been taken to be 0.77, and the value of $crossover_{prob}$ has been taken to be 0.96 in all the runs. The value of $mutation_{prob}$ has been decided empirically by taking several runs of the proposed algorithm. The value of $crossover_{prob}$ has been intentionally kept high to ensure that most members of the target population undergo recombination because this not only facilitates increased convergence to the optimal solution but also ensures that diversity is improved. The results are as follows:

FOR $i = 1; i \le n$

Search for i in the ***parent chromosome***;

$current\_gene = i;$

FOR $j = 1; j \le n;$

Search for $x_{right}(j);$ // $x_{right}(j)$ is the first element to the right of j which has not occurred in the list.

Search for $x_{left}(j);$ // $x_{left}(j)$ is the first element to the left of j which has not occurred in the list.

$p_R (\in [0,1])$ and $p_L (\in [0,1])$ are generated randomly.

IF $p_R > p_L$

$x_{right}(j)$ is inserted immediately after j ;

$current\_gene = x_{right}(j);$

ELSE

$x_{left}(j)$ is inserted immediately after j ;

$current\_gene = x_{left}(j);$

END IF

END FOR

END FOR

Out of the n chromosomes generated, let p' represent the fittest one.

IF $f(p') \ge f(p)$

p is replaced by p'.

CONTINUE GRLS

ELSE

ABORT GRLS

END IF

Fig. 2. The pseudo-code for the GRLS

```
procedure Hybrid Discrete Differential Evolution(HDDE)
initialize parameters
initialize target population
evaluate target population
apply local search(Greedy Reordering Operator)
generation=0;
/* G is defined by the user */
while (generation<G)
obtain the mutant population
obtain the trial population
evaluate the trial population
Select the chromosomes that may progress to the next generation
Apply Local Search(GRLS operator)
generation=generation+1;
end while
Return the best found solution
end HDDE
```

Fig. 3. The pseudo-code for the Hybrid Discrete Differential Evolution(HDDE)

Table 1. Mallya's 5-item problem (Average Daily Cost)

HDDE	Hybrid GA proposed by Moon *et al*.[10]	Dobson's Heuristic[18]
59.87	60.91	61.83

Optimal Production sequence for Mallya's 5-item problem (as per the HDDE):

(4 , 5 , 3 , 1 , 3 , 2 , 3 , 4 , 3 , 1 , 2)

Table 2. Bomberger's 10-item Problem (Average Daily Cost)

HDDE	Hybrid GA proposed by Moon *et al*.[10]	Dobson's Heuristic[18]	Common Cycle Solution[3]	Khouza's GA [17]
125.28	126.12	128.43	231.44	196.14

Optimal Production sequence for Bomberger's 10-item problem (as per the HDDE):

(8 , 9 , 6 , 8 , 3 , 8 , 4 , 2 , 4 , 8 , 9 , 3 , 8 , 10 , 5 , 10 , 8 , 4 , 3 , 2 , 8 , 5 , 9 , 8 , 5 , 8 , 4 , 1 , 8 , 6 , 2 , 8 , 5 , 2 , 7 , 8 , 9 , 4 , 3 , 4 , 5 , 4)

In this paper, we have compared the HDDE that we proposed, with other algorithms that belong to the time varying lot sizes approach in the literature. We have used the data for Mallya's 5-item problem and Bomberger's 10-item problem (for the μ =0.01 case which represents a highly loaded facility). As it can be clearly seen, our algorithm clearly outperforms the hybrid GA proposed by Moon *et al.* by 1.7% and 0.66%, in case of Mallya's 5-item and Bomberger's 10-item problem, respectively.

4 Conclusions and Further Work

This paper has proposed a new type of Hybrid Discrete Differential Evolution algorithm that employs a novel Greedy Reordering operator for local search. We have compared our algorithm with those proposed by Dobson, Khouza and Moon on the Mallya's 5-item and Bomberger's 10-item problems (for the μ=0.01 case) and it can be clearly seen that our algorithms clearly outperforms the rest. Since the ELSP is a very complex combinatorial optimization algorithm, most researchers have developed heuristics that may solve this algorithm efficiently. We have used a heuristics to generate the frequencies and a HDDE (coupled with a local search operator) to find the fittest ELSP sequence which is feasible. Hence, using this algorithm we have increased the speed and accuracy of the search process. Furthermore, this algorithm may be modified accordingly to solve the ELSP with remanufacturing model.

References

1. Hsu, W.: On the general feasibility test of scheduling lot sizes for several products on one machine. Management Science 29, 93–105 (1983)
2. Zanoni, S., Segerstedt, A., Tang, O., Mazzoldi, L.: Multi-product economic lot scheduling problem with manufacturing and remanufacturing using a basic policy period. Computers and Industrial Engineering 62, 1025–1033 (2012)
3. Hanssmann, F.: Operations Research in Production and Inventory. Wiley, New York (1962)
4. Tasgeterin, M.F., Bulut, O., Fadiloglu, M.M.: A discrete artificial bee colony for the economic lot scheduling problem. In: IEEE Congress on Evolutionary Computing (CEC), New Orleans, USA, pp. 347–353 (2012)
5. Dobson, G.: The economic lot scheduling problem: achieving feasibility using time-varying lot sizes. Operations Research 35, 764–771 (1987)
6. Grznar, J., Riggle, C.: An optimal algorithm for the basic period approach to the economic lot schedule problem. Omega 25, 355–364 (1997)
7. Sun, H., Huang, H., Jaruphongsa, W.: The economic lot scheduling problem under extended basic period and power-of-two policy. Optimization Letters 4, 157–172 (2010)
8. Maxwell, W.: The scheduling of economic lot sizes. Naval Research Logistics Quarterly 11, 89–124 (1964)
9. Bomberger, E.: A dynamic programming approach to a lot size scheduling problem. Management Science 12, 778–784 (1966)
10. Moon, I., Silver, E.A., Choi, S.: Hybrid Genetic Algorithm for the Economic Lot Scheduling Problem. International Journal of Production Research 40(4), 809–824 (2002)
11. Zipkin, P.H.: Computing optimal lot sizes in the economic lot scheduling problem. Operations Research 39(1), 56–63 (1991)
12. Dagli, C., Sittisathanchai, S.: Genetic neuro-scheduler for job shop scheduling. International Journal of Production Economics 41(1-3), 135–145 (1993)
13. Grefenstette, J.J.: Incorporating problem specific knowledge into genetic algorithms. In: Davis, L. (ed.) Genetic Algorithms and Simulated Annealing, pp. 42–60. Morgan Kaufmann, Los Altos (1987)
14. Das, S., Suganthan, P.N.: Differential evolution: a strategy of the state of the art. IEEE Transactions on Evolutionary Computation 15(1), 4–31 (2011)

15. Pan, Q.K., Tasgetiren, M.F., Liang, Y.: A discrete differential evolution algorithm for the discrete flow-shop scheduling problem. Computers and Industrial Engineering 55, 795–816 (2007)
16. Mallya, R.: Multi-product scheduling on a single machine: a case study. Omega 20, 529–534 (1992)
17. Khouza, M., Michalewicz, Z., Wilmot, M.: The use of genetic algorithms to solve the economic lot size scheduling problem. European Journal of Operational Research 110, 509–524 (1998)
18. Dobson, G.: The cyclic lot scheduling problem with sequence-dependent setups. Operations Research 40, 736–749 (1992)
19. Ouyang, H., Zhu, X.: An economic lot scheduling problem for manufacturing and remanufacturing. In: IEEE Conference on Cybernetics and Intelligent Systems, Chengdu, pp. 1171–1175 (2008)