Hybridization of Multiple Intelligent Schemes to Solve Economic Lot Scheduling Problem Using Basic Period Approach

Syed Hasan Adil [1], Syed Saad Azhar Ali [2], Aarij Hussaan [1], Kamran Raza [1]

[1.] Department of Computer Science, [2.] Department of Electronic Engineering, Iqra University,
Main Campus: Defence View, Shaheed-e-Millat Road (Ext.) Karachi-75500, Pakistan
hasan.adil@iqra.edu.pk

Abstract: Economic Lot Scheduling Problem (ELSP) has been an area of active research for many years. Different approaches have been proposed to find the optimal solution for the problem. Traditionally, researchers have used a single algorithm to find the solution. In this paper, we argue that better results can be obtained for the ELSP problem, if we use a hybridization scheme instead of the traditional single algorithm approach. In this context, we suggest multiple hybridization of an "intelligent" technique with Golden Section Search (GSS) to solve ELSP using basic period approach. We have used three hybrid approaches based on Simulated Annealing (SA), Cuckoo Search (CS), and Particle Swarm Optimization (PSO) to find the optimum value of integer multiple k_i*'s* and GSS to find the optimum value of basic period *T*. The proposed hybridized schemes are applied on Bomberger's dataset [1], random data generated using distribution given in Dobson's [2] and also on random data generated using new distribution derived from Bomberger's dataset [1]. Comparative analyses are presented in which the hybridized algorithms based on SA, CS and PSO incorporated with GSS are compared. These hybridized schemes were found efficient for both low and high machine utilization.

[Adil SH, Ali SSA, Hussaan A, Raza K. **Hybridization of Multiple Intelligent Schemes to Solve Economic Lot Scheduling Problem Using Basic Period Approach**. *Life Sci J* 2013;10(2):2992-3005] (ISSN:1097-8135). http://www.lifesciencesite.com. 414

Keywords: Economic Lot Scheduling Problem; Basic Period Approach; Cuckoo Search; Particle Swarm Optimization; Golden Section Search.

1. Introduction

The ELSP has been under research for more than four decades. The problem is computationally very complex and has been classified as NP-hard problem [1]. Despite its complexity the ELSP has been encountered in most production planning scenarios [3]. Due to the NP hard nature of the problem many researchers have developed heuristic solutions to the problem. There are four approaches to solve the ELSP problem: common cycle [4]; basic period [5]; extended basic approach [6]; and time varying lot size approach [2].

As the ELSP is generally viewed as NP-hard, the focus of most research efforts has been towards generating near optimal repetitive schedule(s). To date, several heuristic solutions [5, 7, 10, 11, 12, 13, 14, 15, 18, 19] have been proposed using any one of the common cycle, basic period, extended basic approach, or time-varying lot size approaches. The common cycle approach always produces a feasible schedule and is the simplest to implement, however, in some cases the solution when compared to the lower bound is of poor quality [16]. Unlike the common cycle approach, the basic period approach allows different cycle times for different products, however, the cycle times must be an integer multiple of a basic period. Although the basic period approach generally produces a better solution to ELSP than common cycle approach, but getting a feasible schedule is NP-hard [1]. The basic period approach assumes that the production runs of all products shall be made in each basic period. Therefore, the basic period must be long enough to accommodate the production of all the products. This is a rather restrictive condition which usually results in suboptimal solutions. The extended basic period approach removes this restriction and admits the possibility that in any basic period only a subset of the products shall be produced [17, 18]. This obviates the waste of capacity of the production facility. Lastly, the time-varying lot size approach allows for different lot sizes for the different products in a cycle [16]. Dobson [2] showed that the time-varying lot size approach always produced a feasible schedule.

The proposed research is motivated by the recent success [3, 5, 8, 9, 10, 22, 25, 26, 27] of the meta-heuristics to solve ELSP. Therefore, this research investigates the use of meta-heuristics to solve the ELSP problem using basic period approach. We applied PSO, CS, and SA to find the solution. The meta-heuristics will be compared in order to calibrate their performance in regards to solution quality produced and computation time needed.

The rest of the paper is organized as follows: Section 2 outlines the problem statement. Section 3 describes the theory behind basic period approach to

ELSP. Section 4 describes the proposed hybrid approach. Section 5 gives an introduction to the GSS algorithm. Section 6 gives the reason behind the selection of hybridization of intelligent techniques with GSS. Section 7 gives an introduction to the PSO algorithm and our proposed hybridization scheme using PSO and GSS. Similarly, Section 8 gives an introduction CS algorithm and our proposed hybridization scheme using CS and GSS. Section 9 presents our hybridization scheme using SA and GSS. In Section 10, we compare the results of our proposed approaches with other results. We present our discussions and conclusions in Section 11.

2. Problem Statement

The ELSP is to schedule the production of several different items in the same facility on repetitive basis. The facility is such that only one item can be produced at a time, there is a setup cost and a setup time associated with each item, the demand rate for each item is constant over an infinite planning horizon, and no shortages are allowed [1, 5]. A feasible production schedule is defined as the one in which: (a) at most one item is produced by the facility at any time (b) the total time load on the facility does not exceed the available time capacity; and (c) demand is satisfied without shortages.

3. Basic Period Approach to ELSP

We present ELSP model [1] which is based on the basic period approach. We have to produce *m* distinct products on single production facility with the following assumptions.

- The competing products for production facility do not have any precedence over each other.
- Back-orders are not allowed.
- An item is considered for production only if its inventory is depleted to the zero level. This rule is known as Zero-Switching-Rule (ZSR).
- The production facility is assumed to be failure free and to always produce perfect quality products.

The solution of the ELSP is based on specifying an inventory cycle for each part, subject to following conditions:

- The quantity of a part produced during its cycle must be sufficient to meet demand over the cycle.
- The length of the cycle must be sufficient to permit the production of other parts scheduled during the cycle.

A schedule is feasible if the above conditions are met. This feasible solution becomes optimal if the total cost minimizes.

The following notations and equations (1-14) are used to find the solution of ELSP [1, 5]:

i	:	An item index, $i=\{1,2,\ldots,n\}$
D_i	:	Annual demand for item i (units/ year)
P_i	:	Annual production rate for item i (units/year)
H_i	:	Holding cost for item i (\$/unit-year)
S_i	:	Setup cost for item i (\$/setup)
τ_i	:	Setup time for item i (years)
Q_i	:	Production quantity for item i, a decision variable (units)
T_i	:	Cycle time for item i, a decision variable (in days)
TC_i	:	Total annual holding and setup cost for item i (\$/year)
TC	:	Total annual holding and setup cost for all item (\$/year)

The total cost for an item i is:

$$TC_i = \frac{Q_i}{2}\left(1 - {}^{D_i}\!/_{P_i}\right) H_i + \left({}^{D_i}\!/_{Q_i}\right) S_i \qquad (1)$$

The total annual cost of all n items is:

$$TC = \sum_{i=1}^{n}\left[\frac{Q_i}{2}\left(1 - {}^{D_i}\!/_{P_i}\right) H_i + \left({}^{D_i}\!/_{Q_i}\right) S_i\right] \qquad (2)$$

The ELSP is formulated as follows:
Minimize TC

$$\text{Subject to} \quad \sum_{i=1}^{n}\left(\left({}^{D_i}\!/_{P_i}\right)\tau_i + \frac{D_i}{P_i}\right) \le 1 \qquad (3)$$

No two items are produced at the same time (4)

The first constraint ensures that the time spent setting up the machine and producing the items does not exceed the time available. Solving the unconstrained problem results a loose lower bound known as the independent solution (IS). The optimal order quantity for item i is:

$$Q_i^* = \sqrt{\frac{2\,D_i S_i P_i}{H_i\left(1 - {}^{D_i}\!/_{P_i}\right)}} \qquad (5)$$

Substituting from equation (5) into equation (2) gives IS lower bound on the ELSP as follows:

$$TCIS = \sum_{i=1}^{n}\sqrt{\frac{2\,D_i\,S_i\,H_i}{(P_i - D_i)P_i}} \qquad (6)$$

Alternatively, a tighter lower bound (TCL) can be obtained by minimizing the total cost (TC) subject to constraint in equation (3):

$$Q_i^* = \sqrt{\frac{2\,D_i\,P_i(S_i + \lambda\,\tau_i)}{H_i(P_i - D_i)}} \qquad (7)$$

And satisfying:

$$\lambda\left(\sum_{i=1}^{n}\frac{\tau_i D_i}{Q_i}+\sum_{i=1}^{n}\frac{D_i}{P_i}-1\right)=0 \qquad (8)$$

In case if the production facility in under-utilized, the capacity constraint will not be binding and TCL will be same as TCIS. However, with the higher utilization, TCL is higher than the IS lower bound. The increase in TC and TCL relative to TCIS at high utilization is due to production quantities becoming larger to reduce the time spend on setup, which substantially increases the holding cost.

Now, we discuss an analytical approach which allows achieving the optimal solution to a restricted version of the original problem mentioned in [6, 19]. The approach is called basic period approach. In basic period approach, the cycle time for every item *i* is an integer multiple k_i of a fundamental cycle *T*. Thus, the cycle time for an item *i* is:

$$T_i = k_i T \qquad (9)$$

Also the production quantity for an item i will becomes:

$$Q_i = T_i D \qquad (10)$$

The total cost incurred under basic period approach (TCBP) is obtained from substituting T_i and Q_i into equation (2). Thus, the total cost is:

$$TCBP=\sum_{i=1}^{n} T k_i D_i\left(1-\frac{D_i}{P_i}\right)\frac{H_i}{2}+\frac{S_i}{T\,k_i} \qquad (11)$$

TCBP established in Equation (11) is now a function of *T* and k_i*'s*. Once TCBP is established, the ELSP under BP approach is:

Minimize TCBP

$$\text{Subject to} \quad \sum_{i=1}^{n}\left(\tau_i+\frac{D_i\,T\,k_i}{P_i}\right)\le T \qquad (12)$$

The constraint in the above optimization problem ensures that the fundamental cycle is long enough to accommodate the production of all items even though not every item has to be produced during every fundamental cycle. The constraint guarantees the feasibility but may result in a suboptimal solution to the original problem. In [1], it is shown that the above problem can be formulated and solved as a Dynamic Programming (DP) problem. The main idea of [1] was to fix *T*, and solve the DP problem to obtain the optimal k_i*'s* and then use the information to get a better estimate of the optimal *T*. Thus, this approach requires solving a number of DP problems to find the optimal *T*.

In a nutshell this approach requires a one-dimensional search on *T*. In each of the iteration of the search, a DP problem must be solved. Thus, a more precise estimate of the optimal *T* requires larger number of the DP problems to be solved that makes the use of meta-heuristics even more attractive alternate to solve the problem. The above formulation very well suits meta-heuristics. GA [5] suggested that both the *T* and k_i*'s* are simultaneously determined leaving no need to solve DP problems repeatedly with different values of *T*. Furthermore, the curse of dimensionality due to DP is not encountered in using GA.

4. Proposed Hybridized Approach

In this research; we suggest multiple hybridization of an “intelligent” technique with GSS to solve ELSP using basic period approach. We have used three hybrid approaches based on SA, CS, and PSO to find the optimum value of integer multiple k_i*'s* and GSS to find the optimum value of basic period *T*. The proposed hybridized schemes are analyzed using Bomberger’s dataset [1], random data generated using distribution given in Dobson [2], and random data generated using new distribution derived from Bomberger’s dataset [1].

5. Golden Section Search

GSS [20] is an optimization technique that finds the optimum (i.e., minimum/maximum) of a function in one dimensional search space. In order to understand the working of GSS algorithm, we first need to understand Bisection Method (BM) for finding root of a function. Given an interval [a, b] such that *f(a)* * *f(b)*< 0 and also function is continuous in the given interval, BM finds the root of a function in an iterative manner by first computing the midpoint *m* of the interval [a, b] so that we have two intervals [a, m] and [m, b], it then selects the interval which is closer to the root of the function. The BM algorithm will repeat the same procedure until *f(m)* = 0 or abs(*b* - *a*)< tolerance value (i.e., abs() function will always give positive value).

GSS algorithm is also similar to BM algorithm. We first need to provide an interval [a, c] in which we want to find the minimum of the function (i.e. for maximum we can just take the negative of the function). GSS is only able to find minimum of the function if we have a triplet of points $a<b<c$, such that $f(b)<f(a)$ and $f(b)<f(c)$. In this case we are sure that the function (if it is smooth) has a minimum in the interval [a, c].

The basic working of the GSS can be described as follows:

- Given an interval [a, c], GSS first bracket the minimum of the function with a triplet $a<b<c$, such that $f(b)<f(a)$ and $f(b)<f(c)$.

- The optimal bracketing interval (a, b, c) has its middle b a fractional distance 0.38197 from one end (say a), and 0.61803 from the other end (say b). These fractions are known as golden mean or golden section.
- Repeat the following step until the minimum of the function converged to the desired tolerance level.
- Using the current bracketing triplet of points, the next point to be tried is a fraction 0.38197 into the larger of the two intervals (measuring from the central point of the triplet). If it starts out with a bracketing triplet whose segments are not in the golden ratios, the procedure of choosing successive points at the golden mean point of the larger segment will quickly converge to the proper self-replicating ratios.

6. Why Hybridization with Golden Section Search

In this paper three nature inspired optimization techniques including SA, PSO, and CS are hybridized with GSS technique to find the optimum value for integer value k_i's and basic Period T respectively. In this proposed technique we first find the value of k_i's using SA/CS/PSO and then used these values to find the value of T. It is important to observe that for a given value of k_i's the ELSP cost function becomes one variable uni-modal function as shown in Fig 1. For one variable uni-modal scenario, we don't need to apply any complex optimization technique instead we can apply GSS to efficiently find the minima of the function (i.e., the value of T where cost is minimum).

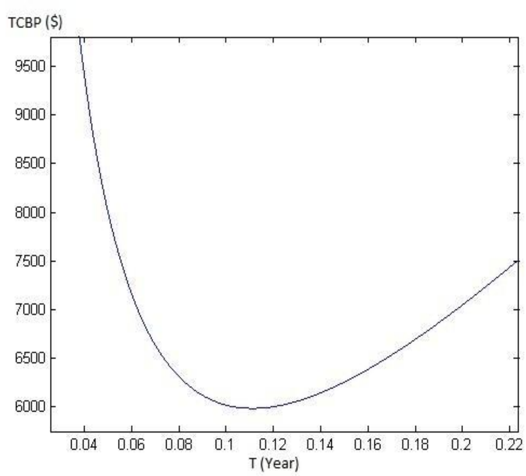

Figure 1. ELSP cost function for specific k_i's

7. Particle Swarm Optimization

Particle swarm optimization is a population based swarm intelligence algorithm. It was originally proposed by Kennedy [26] as a simulation of the social behavior of social organisms, such as bird flocking and fish schooling. PSO uses the physical movements of the individuals (particles) in the swarm and has a flexible and well balanced mechanism to enhance and adapt to global and local exploration abilities. The PSO algorithm is widely used in many optimization problems due to the intrinsic simplicity of the algorithm itself. It does not require mathematical computation like derivatives or complex encoding like Genetic Algorithm. PSO maintain best solution of each particle along with the global best solution of the whole population and therefore it is less sensitive to local minima problem.

The PSO algorithm works by selecting a set of P particles and initialized by placing it into random positions in the solution space. The position of each particle represents a solution to the problem and its performance is evaluated by objective function specific to a particular problem. The velocity of the each particle v_j is defined as the change of its position. The direction of movement of each particle is the active interaction of individual and whole swarm flying experiences. Each particle adjusts its path towards the solution based on its own previous best position and previous best position of the whole population, namely p_j and p_g. The velocities and positions of particles are updated using the following formulas:

$$v_j(t+1) = v_j(t) + c_j rand_j\left(p_j - s_j(t)\right) + c_g rand_g\left(p_g - s_j(t)\right) \quad (13)$$

$$s_j(t+1) = s_j(t) + v_j(t+1) \quad (14)$$

Where t is the previous iteration and t+1 is the current iteration to compute; c_j and c_g are the acceleration coefficients; $rand_j$, $rand_g$ are random numbers between 0 and 1 inclusive associated with the best solution of a particular particle and the best solution of the whole swarm. c_j and c_g are used to provide the maximum distance a particle will move in a single iteration. The objective function is than computed using particles placed in new positions at iteration t+1. The same equations (13) and (14) are repeated until the maximum iteration becomes reached or until a convergence criterion has been met. At the end of all iterations the best solution found by the whole swarm is returned.

A. Proposed GSS-PSO Hybridization Scheme

The proposed hybridized PSO with GSS algorithm is discussed below:

- The nonlinear objective function given in equation (11) is minimized subject to constraint given in equation (12).
- Computes lower and upper bounds of T and k_i's using following equations [5],

$$T^{LB} = \sum_{i=1}^{n} 0.25\, Q_i^*/P_i \qquad (15)$$

$$T^{UB} = \max\left\{ \begin{matrix} \sqrt{(2(\sum_{i=1}^{n} S_i)/\sum_{i=1}^{n} H_i D_i (1 - D_i P_i)),} \\ (\sum_{i=1}^{n} \tau_i)/(1 - D_i P_i) \end{matrix} \right\} \qquad (16)$$

$$k_i^{LB} = 1 \qquad (17)$$

$$k_i^{UB} = \left\lceil (5\,(Q_i^*/D_i)/T)(\sum_{i=1}^{n} \frac{D_i}{P_i}) \right\rceil, i = 1, 2, \dots, n \qquad (18)$$

- Initializes k_i*'s* randomly between $[k_i^{LB}, k_i^{UB}]$, $i = 1, 2, \dots, n$
- Given the initial k_i*'s*, the TCBP subject to constraint (12) can be minimized by performing a one dimensional search on T based on GSS as discussed in Section 5 [20].
- Repeat the following steps until the maximum iteration becomes reached or until a convergence criterion has been met.
- Apply PSO algorithm as earlier using equation (13) and (14) to generate the new positions of P particles in k-dimensional search space. Here, the position of each particle in each dimension represents one k_i and the whole particle represents one complete possible solution to ELSP problem
- Updates k_i*'s* associated with each particle that do not fulfill lower and upper bound requirements with randomly generated values between $[k_i^{LB}, k_i^{UB}]$.
- Given newly generated k_i*'s* associated with each particle in k-dimensional search space; apply a one dimensional search on T based on GSS as discussed in section V [20] to minimize TCBP subject to constraint (12).
- Updates current best k_i*'s* and T that minimize TCBP.
- Updates best position (solution) p_j of each particle in the swarm.
- Updates best position p_g of the whole swarm using best solution of all the particles in the swarm.

8. Cuckoo Search Optimization

CS is a population based optimization algorithm. It was originally proposed by Yang [21] for solving optimization problems. CS is based on the obligate brood parasitic behavior of some cuckoo species in combination with the Lévy Flight behavior of some birds and fruit flies. The CS is comparatively simpler than other meta-heuristic techniques. During each iteration CS computes fitness function and based on the output worst nests are abandoned (i.e., nest which does not provide good solution). In each generation CS moves towards global optimum by replacing the possible solutions with the good ones and at the end of the execution optimum solution is obtained.

The basic working of the CS algorithm can be described as follows:

- Initializes N random host nest.
- The number of available host nests is fixed.
- Each cuckoo lays one egg at a time and dumps it in a randomly chosen nest.
- Generates new N nests using the Lévy Walk around the best solution obtained so far this will speed up the local search.
- Compares old nests with corresponding new nests and selects the best N nests from them.
- A host can discover an alien egg with a probability p_i. If the $p_i > p_a$ (i.e., p_a is the probability of discovering alien eggs) then the host bird can either throw the egg away or abandon the nest.
- For abandoned nests CS generates new random nests having locations far enough from the current best solution. This will make sure the system will not be trapped in a local optimum.
- The best nests with high quality of eggs (i.e. solutions) will be carried over to the next generations.

A. Proposed GSS-CS Hybridization Scheme

The proposed hybridized CS with GSS algorithm is discussed below:

- The nonlinear objective function given in equation (11) is minimized subject to constraint given in equation (12).
- Computes lower and upper bounds of T and k_i*'s* using equations (13, 14, 15, 16),
- Initializes k_i*'s* randomly between $[k_i^{LB}, k_i^{UB}]$, $i = 1, 2, \dots, n$
- Given the initial k_i*'s*, the TCBP subject to constraint (12) can be minimized by performing a one dimensional search on T based on GSS as discussed in Section 5 [20].
- Repeat the following steps until the maximum number of iteration is reached or until a convergence criterion is met.
- Apply CS algorithm as discussed earlier to generate/select N nests in k-dimensional search space. Here, the position of each nest in each dimension represents one k_i and the whole nest

represents one complete possible solution to ELSP problem

- Updates k_i's associated with each particle that do not fulfill lower and upper bound requirements with randomly generated values between $[k_i^{LB}, k_i^{UB}]$.
- Given newly generated k_i's associated with each nest in k-dimensional search space; apply a one dimensional search on T based on GSS as discussed in Section 5 [20] to minimize TCBP subject to constraint (12).
- Updates current best k_i's and T that minimize TCBP

9. Simulated Annealing Optimization

Simulated annealing is a popular meta-heuristic algorithm for addressing optimization problems. The highlighting factor of this algorithm is its ability to escape the local optima by broadening its search area in order to find the global optimum. It derives its name from the physical process of annealing with solid ores, where the crystalline solids are heated and then they are cooled slowly until they achieve a configuration of crystals free of defects. Simulated Annealing uses these principles to search for global optimums of optimization problems. The basic pseudo-code [23, 24] of this algorithm is shown below:

Select an initial solution $\omega \epsilon \Omega$
Select the temperature change counter k=0
Select a temperature cooling schedule, t_k
Select an initial temperature T = t_o >= 0
Select a repetition schedule M_k that defines the number of iterations executed at each temperature t_k
Repeat
Set repetition counter m = 0
    Repeat
        Generate a solution $\omega' \in N(\omega)$
        Calculate $\Delta_{\omega\omega'} = f(\omega') - f(\omega)$
        If $\Delta_{\omega\omega'} \leq 0$ *then* $\omega \leftarrow \omega'$
        If $\Delta_{\omega\omega'} > 0$ *then* $\omega \leftarrow \omega'$ with probability $\exp(-\Delta_{\omega\omega'}/t_k)$
        $m \leftarrow m + 1$
    Until $m = M_k$
    $k \leftarrow k + 1$
Until stopping criterion is met.

A. Proposed GSS-SA Hybridization Scheme

The proposed hybridized SA with GSS algorithm is discussed below:

- The nonlinear objective function given in equation (11) is minimized subject to constraint given in equation (12).
- Computes lower and upper bounds of T and k_i's using equations (13, 14, 15, 16),
- Initializes k_i's randomly between $[k_i^{LB}, k_i^{UB}]$, i = 1, 2, …, n
- Given the initial k_i's, the TCBP subject to constraint (12) can be minimized by performing a one dimensional search on T based on GSS as discussed in section V [20].
- Repeat the following steps until the maximum number of iteration is reached or until a convergence criterion is met.
- Apply SA algorithm as discussed earlier to generate/select metropolis in k-dimensional search space. Here, the position of metropolis in each dimension represents one k_i and the whole nest represents one complete possible solution to ELSP problem
- Updates k_i's associated with each particle that do not fulfill lower and upper bound requirements with randomly generated values between $[k_i^{LB}, k_i^{UB}]$.
- Given newly generated k_i's associated with metropolis in k-dimensional search space; apply a one dimensional search on T based on GSS as discussed in Section 5 [20] to minimize TCBP subject to constraint (12).
- Updates current best k_i's and T that minimize TCBP.

10. Results

In this study we performed three different computational analysis using SA, CS, and PSO. First analysis is based on [1] dataset as shown in Table 1, second analysis is based on random data generated using three distribution given in [2] as shown in Table 2, and the third analysis is based on random data generated using [1] as shown in Table 3.

A. Numerical Experiment 1

The results obtained from first analysis are shown in Table 4, Table 5, Table 6, and Table 7. Table 4 compares the cost obtained by solving [1] problem using SA, CS, PSO and GA [5] algorithms. Table 5 compares the (i) relative deviation from tighter lower bound (TCL), (ii) improvement achieved through SA, CS, and PSO algorithms over results obtained through GA algorithm [5], (iii) efficiency in terms of execution time taken by TS, SA, CS, and PSO algorithms. Table 6 and Table 7 compare the detailed solution found by CS, and PSO with GA solution [5].

Table 1: Data of Bomberger's problem.

Product index, i	1	2	3	4	5	6	7	8	9	10
Base Demand	24,000	24,000	48,000	96,000	4800	4800	1440	20,400	20,400	24,000
Setup cost (S_i): $	15	20	30	10	110	50	310	130	200	5
Production rate (P_i): units/day	30,000	8000	9500	7500	2000	6000	2400	1300	2000	15,000
Setup time (τ_i) : h	1	1	2	1	4	2	8	4	6	1
Holding cost (H_i): $/unit-year	0.00065	0.01775	0.01275	0.01000	0.27850	0.02675	0.15000	0.59000	0.09000	0.00400

Table 2: Distribution for randomly generated data by Dobson [2].

Parameters	Set 1	Set 2	Set 3
Number of items (units)	[5, 15]	[5, 15]	[5, 15]
Production rate (units/unit-time)	[2000, 20000]	[4000, 20000]	[1500, 30000]
Demand rate (units/unit-time)	[1500, 2000]	[1000, 2000]	[500, 2000]
Set-up time (time/unit)	[1, 4]	[1, 4]	[1, 8]
Setup cost ($)	[50, 100]	[50, 100]	[10, 350]
Holding cost ($)	[1/240, 6/240]	[1/240, 6/240]	[5/240000, 5/240]

Table 3: Distribution for randomly generated data using Bomberger's problem [1].

Parameters	Range
Number of items (units)	[10, 30]
Production rate (units/unit-time)	[31,2000, 720,0000]
Demand rate (units/unit-time)	[1440, 96,000]
Set-up time (time/unit)	[1/1920, 8/1920]
Setup cost ($)	[5, 310]
Holding cost ($)	[0.00065, 0.59000]

Table 4 shows that 77% of CS solutions are either better or similar to best result obtained from any other algorithm, 71% of PSO solutions are either better or similar to best result obtained from any other algorithm, 48% of SA solutions are either better or similar to best result obtained from any other algorithm, while only 41% of GA solution are better or similar to best result obtained from any other algorithm. So, in majority of cases CS performed better than all other algorithms. Table 5 shows that the best average relative deviation from TCL is 19.542% using CS and worst average relative deviation from TCL is 21.261% using GA algorithm. Best average improvement over GA is 0.966% using CS, and best average CPU utilization time is 5.192 sec using SA. It is also important to note that CS, and PSO all have same relative deviation from TCL for high utilization and only differs in low utilization cases. However, GA differs with other algorithms for high utilization as well as low utilization cases. GA found worst relative deviation from TCL for higher utilization but results for lower utilization cases are comparatively closed to other algorithms.

Table 6 shows the detail comparison of values for T and k_i (i.e., i=1,2,…10) using GA and SA algorithm, Table 7 shows the detail comparison of values for T and k_i (i.e., i=1,2,…10) using GA and CS algorithm, while Table 8 shows the detailed comparison of values obtained for T and k_i using GA and PSO algorithm. For low utilization cases 50 to 92 k_i have different values but for high utilization cases 95 to 99 all k_i have same value '1'. CS and PSO found same value for T and k_i which gives low deviation from TCL. GA found the same value for k_i but failed to found value for T similar to other algorithms and therefore it results in high deviation from TCL.

Table 4: Comparison of TSIS, TCL, GA, SA, CS, and PSO solutions for Bomberger's problem [1, 5].

Utilization (%)	TSIS	TCL	GA	SA	CS	PSO	Best Cost	Best Algorithm(s)
50	5960.445	5960.445	6038.410	6036.513	6032.225	6036.513	6032.225	CS
55	6218.253	6218.253	6328.670	6372.022	6328.086	6328.086	6328.086	CS,PSO
60	6459.905	6459.905	6621.750	6619.799	6618.572	6618.572	6618.572	CS,PSO
65	6687.131	6687.131	6914.700	6914.837	6914.837	6914.837	6914.700	GA
66.18	6738.810	6738.810	7024.110	7124.987	7024.100	7024.100	7024.100	GA,CS,PSO
70	6901.335	6901.335	7395.460	7395.467	7395.466	7395.466	7395.460	All
75	7103.674	7103.674	7789.630	7917.524	7794.202	7794.202	7789.630	GA
80	7295.114	7295.114	8096.010	8181.051	8085.485	8085.485	8085.485	CS,PSO
83	7405.090	7405.090	8250.290	8250.290	8250.290	8250.290	8250.290	GA,SA,CS,PSO
86	7511.593	7511.593	8553.310	8483.945	8483.945	8483.945	8483.945	SA,CS,PSO
88.24	7588.934	7588.934	8782.420	8782.289	8782.289	8782.289	8782.289	SA,CS,PSO
89	7614.763	7614.763	8874.550	8874.803	8874.803	8874.803	8874.550	GA
92	7714.729	7714.729	9745.800	9746.356	9746.356	10086.443	9745.800	GA
95	7811.608	8418.885	12018.080	11949.646	11949.646	11949.646	11949.646	SA,CS,PSO
97	7874.534	11290.966	17143.000	17134.260	17134.260	17134.260	17134.260	SA,CS,PSO
98	7905.510	15681.535	24533.820	24457.541	24457.541	24457.541	24457.541	SA,CS,PSO
99	7936.166	29942.667	55544.470	47550.735	47550.735	47550.735	47550.735	SA,CS,PSO

Table 5: Comparison of Relative Deviation from TCL, Improvement over GA, and CPU time taken by algorithms for Bomberger's problem [1, 5].

	% Relative Deviation from TCL				% Improvement over GA			CPU time (sec.)		
Utilization (%)	GA	SA	CS	PSO	SA	CS	PSO	SA	CS	PSO
50	1.308	1.276	1.204	1.276	0.031	0.102	0.031	7.281	12.852	15.189
55	1.776	2.473	1.766	1.766	0	0.009	0.009	7.178	12.572	15.019
60	2.505	2.475	2.456	2.456	0.029	0.048	0.048	7.409	12.533	15.489
65	3.403	3.405	3.405	3.405	0	0	0	7.731	12.789	15.690
66.18	4.234	5.731	4.234	4.234	0	0	0	5.81	12.829	15.972
70	7.160	7.160	7.160	7.160	0	0	0	4.946	12.544	16.059
75	9.656	11.457	9.721	9.721	0	0	0	20.023	12.168	16.060
80	10.979	12.144	10.834	10.834	0	0.130	0.130	2.838	12.184	15.561
83	11.414	11.414	11.414	11.414	0	0	0	2.815	12.274	16.205
86	13.868	12.945	12.945	12.945	0.811	0.811	0.811	2.686	12.291	14.813
88.24	15.727	15.725	15.725	15.725	0.001	0.001	0.001	2.791	12.239	13.786
89	16.544	16.547	16.547	16.547	0	0	0	2.742	11.983	13.465
92	26.327	26.334	26.334	30.743	0	0	0	2.788	12.319	11.131
95	42.751	41.939	41.939	41.939	0.569	0.569	0.569	2.553	10.964	11.075
97	51.829	51.752	51.752	51.752	0.051	0.051	0.051	2.71	10.919	11.283
98	56.450	55.964	55.964	55.964	0.311	0.311	0.311	2.753	10.876	11.140
99	85.503	58.806	58.806	58.806	14.392	14.392	14.392	3.205	10.890	11.063
Average	21.261	19.856	19.542	19.805	0.953	0.966	0.962	5.192	12.072	14.059
Min.	1.308	1.276	1.204	1.276	0	0	0	2.553	10.876	11.063
Max.	85.503	58.806	58.806	58.806	14.392	14.392	14.392	20.023	12.852	16.205
Std. Dev.	23.939	19.726	19.923	20.042	3.471	3.467	3.469	4.316	0.706	2.078

Table 6: Detail comparison of GA and SA results for Bomberger's problem [1, 5]

Utilization	Meta-heuristic	
	GA	SA
50	T = 28.183 k_i =[5,1,2,1,2,4,10,1,3,1]	T = 28.594 k_i =[4,1,2,1,2,4,9,1,3,2]
55	T = 28.762 ki =[5,2,2,1,2,4,8,1,2,1]	T = 29.314 k_i =[3,1,1,1,2,4,8,1,3,1]
60	T = 28.863 k_i =[4,1,1,1,2,4,9,1,2,2]	T = 28.798 k_i =[3,2,1,1,2,4,8,1,2,1]
65	T = 30.828 k_i =[2,1,1,1,2,3,7,1,2,1]	T = 30.838 k_i =[2,1,1,1,2,3,7,1,2,1]
66.18	T = 30.443 k_i =[2,1,1,1,2,2,6,1,2,1]	T = 32.422 k_i =[4,1,1,1,1,3,7,1,2,1]
70	T = 33.42 k_i =[2,1,1,1,1,2,3,1,2,1]	T = 33.42 k_i =[2,1,1,1,1,2,5,1,2,1]
75	T = 31.794 k_i =[3,1,1,1,2,3,7,1,1,1]	T = 35.719 k_i =[2,1,1,1,1,2,6,1,1,1]
80	T = 34.438 k_i =[2,1,1,1,1,3,6,1,1,1]	T = 35.614 k_i =[1,1,1,1,1,2,5,1,1,1]
83	T = 34.951 k_i =[1,1,1,1,1,2,5,1,1,1]	T = 34.961 k_i =[2,1,1,1,1,2,5,1,1,1]
86	T = 37.131 k_i =[1,1,1,1,1,1,5,1,1,1]	T = 38.371 k_i =[1,1,1,1,1,2,4,1,1,1]
88.24	T = 38.442 k_i =[1,1,1,1,1,1,3,1,1,1]	T = 38.436 k_i =[1,1,1,1,1,1,3,1,1,1]
89	T = 41.748 k_i =[1,1,1,1,1,1,3,1,1,1]	T = 41.758 k_i =[1,1,1,1,1,1,3,1,1,1]
92	T = 53.904 k_i =[1,1,1,1,1,1,2,1,1,1]	T = 53.914 k_i =[1,1,1,1,1,1,2,1,1,1]
95	T = 75.809 k_i =[1,1,1,1,1,1,1,1,1,1]	T = 75 k_i =[1,1,1,1,1,1,1,1,1,1]
97	T = 125.08 k_i =[1,1,1,1,1,1,1,1,1,1]	T = 125 k_i =[1,1,1,1,1,1,1,1,1,1]
98	T = 188.14 k_i =[1,1,1,1,1,1,1,1,1,1]	T = 187.5 k_i =[1,1,1,1,1,1,1,1,1,1]
99	T = 439.45 k_i =[1,1,1,1,1,1,1,1,1,1]	T= 375 k_i =[1,1,1,1,1,1,1,1,1,1]

Table 7: Detailed result Comparison between GA and CS for Bomberger's problem [1, 5].

Utilization	Meta-heuristic	
	GA	CS
50	T = 28.183 k_i =[5,1,2,1,2,4,10,1,3,1]	T = 28.594 k_i =[3,2,2,1,2,4,8,1,3,1]
55	T = 28.762 k_i =[5,2,2,1,2,4,8,1,2,1]	T = 29.439 k_i =[5,2,2,1,2,4,9,1,2,1]
60	T = 28.863 k_i =[4,1,1,1,2,4,9,1,2,2]	T = 29.306 k_i =[5,1,1,1,2,4,8,1,2,2]
65	T = 30.828 k_i =[2,1,1,1,2,3,7,1,2,1]	T = 30.838 k_i =[2,1,1,1,2,3,7,1,2,1]
66.18	T = 30.443 k_i =[2,1,1,1,2,2,6,1,2,1]	T = 30.449 k_i =[2,1,1,1,2,2,6,1,2,1]
70	T = 33.42 k_i =[2,1,1,1,1,2,3,1,2,1]	T = 33.42 k_i =[2,1,1,1,1,2,5,1,2,1]

75	T = 31.794 k_i =[3,1,1,1,2,3,7,1,1,1]	T = 32.11 k_i =[3,1,1,1,2,4,6,1,1,1]
80	T = 34.438 k_i =[2,1,1,1,1,3,6,1,1,1]	T = 35.28 k_i =[3,1,1,1,1,3,6,1,1,1]
83	T = 34.951 k_i =[1,1,1,1,1,2,5,1,1,1]	T = 34.961 k_i =[2,1,1,1,1,2,5,1,1,1]
86	T = 37.131 k_i =[1,1,1,1,1,1,5,1,1,1]	T = 38.371 k_i =[1,1,1,1,1,2,4,1,1,1]
88.24	T = 38.442 k_i =[1,1,1,1,1,1,3,1,1,1]	T = 38.436 k_i =[1,1,1,1,1,1,3,1,1,1]
89	T = 41.748 k_i =[1,1,1,1,1,1,3,1,1,1]	T = 41.758 k_i =[1,1,1,1,1,1,3,1,1,1]
92	T = 53.904 k_i =[1,1,1,1,1,1,2,1,1,1]	T = 53.914 k_i =[1,1,1,1,1,1,2,1,1,1]
95	T = 75.809 k_i =[1,1,1,1,1,1,1,1,1,1]	T = 75 k_i =[1,1,1,1,1,1,1,1,1,1]
97	T = 125.08 k_i =[1,1,1,1,1,1,1,1,1,1]	T = 125 k_i =[1,1,1,1,1,1,1,1,1,1]
98	T = 188.14 k_i =[1,1,1,1,1,1,1,1,1,1]	T = 187.5 k_i =[1,1,1,1,1,1,1,1,1,1]
99	T = 439.45 k_i =[1,1,1,1,1,1,1,1,1,1]	T = 375 k_i =[1,1,1,1,1,1,1,1,1,1]

Table 8: Detailed result Comparison between GA and PSO for Bomberger's problem [1, 5].

Utilization	Meta-heuristic	
	GA	PSO
50	T = 28.183 k_i =[5,1,2,1,2,4,10,1,3,1]	T = 28.594 k_i =[4,1,2,1,2,4,9,1,3,2]
55	T = 28.762 k_i =[5,2,2,1,2,4,8,1,2,1]	T = 29.439 k_i =[5,2,2,1,2,4,9,1,2,1]
60	T = 28.863 k_i =[4,1,1,1,2,4,9,1,2,2]	T = 29.306 k_i =[5,1,1,1,2,4,8,1,2,2]
65	T = 30.828 k_i =[2,1,1,1,2,3,7,1,2,1]	T = 30.838 k_i =[2,1,1,1,2,3,7,1,2,1]
66.18	T = 30.443 k_i =[2,1,1,1,2,2,6,1,2,1]	T = 30.449 k_i =[2,1,1,1,2,2,6,1,2,1]
70	T = 33.42 k_i =[2,1,1,1,1,2,3,1,2,1]	T = 33.42 k_i =[2,1,1,1,1,2,5,1,2,1]
75	T = 31.794 k_i =[3,1,1,1,2,3,7,1,1,1]	T = 32.11 k_i =[3,1,1,1,2,4,6,1,1,1]
80	T = 34.438 k_i =[2,1,1,1,1,3,6,1,1,1]	T = 35.28 k_i =[3,1,1,1,1,3,6,1,1,1]
83	T = 34.951 k_i =[1,1,1,1,1,2,5,1,1,1]	T = 34.961 k_i =[2,1,1,1,1,2,5,1,1,1]
86	T = 37.131 k_i =[1,1,1,1,1,1,5,1,1,1]	T = 38.371 k_i =[1,1,1,1,1,2,4,1,1,1]
88.24	T = 38.442 k_i =[1,1,1,1,1,1,3,1,1,1]	T = 38.436 k_i =[1,1,1,1,1,1,3,1,1,1]
89	T = 41.748 k_i =[1,1,1,1,1,1,3,1,1,1]	T = 41.758 k_i =[1,1,1,1,1,1,3,1,1,1]
92	T = 53.904 k_i =[1,1,1,1,1,1,2,1,1,1]	T = 46.875 k_i =[1,1,1,1,1,1,1,1,1,1]
95	T = 75.809 k_i =[1,1,1,1,1,1,1,1,1,1]	T = 75 k_i =[1,1,1,1,1,1,1,1,1,1]

97	T = 125.08 k_i =[1,1,1,1,1,1,1,1,1,1]	T = 125 k_i =[1,1,1,1,1,1,1,1,1,1]
98	T = 188.14 k_i =[1,1,1,1,1,1,1,1,1,1]	T = 187.5 k_i =[1,1,1,1,1,1,1,1,1,1]
99	T = 439.45 k_i =[1,1,1,1,1,1,1,1,1,1]	T = 375 k_i =[1,1,1,1,1,1,1,1,1,1]

B. Numerical Experiment 2

The second analysis is based on random data generated from set 1, set 2, and set 3 of [2] using SA, CS and PSO. Table 9 obtained by solving hundred distinct random generated problems of size between five and eight and utilization between 90% and 99% from set 1 of [6] using SA, CS and PSO. The results obtained from all these algorithms are same (i.e., mean deviation from TCL 8.273, minimum deviation from TCL 0.951, maximum deviation from TCL 20.905, and in almost all cases k_i= 1). The higher deviation from TCL is due to the higher utilization factor. We can also refer to the first analysis using [1] problem in which deviation from TCL increases with the increase in utilization (i.e., for 90% utilization relative deviation from TCL was 26.334% and all k_i (i.e., i=1,2,…10) have same value '1').

Table 10 is obtained by solving hundred distinct random data generated for problem size between five and ten from set 2 of Dobson using SA, CS and PSO. The result obtained from all algorithms were same (i.e., mean deviation from TCL 7.154, minimum deviation from TCL 0.834 and maximum deviation from TCL 22.529 and in almost all cases k_i= 1).

Table 11 is obtained by solving hundred distinct random data generated for problem size between five and fifteen from set 3 of [6] using CS and PSO. The result obtained from all these algorithms were same (i.e., mean deviation from TCL 22.308, minimum deviation from TCL 5.347 and maximum deviation from TCL 57.436 and in almost all cases k_i= 1).

Table 9: Comparison of relative deviation from TCL for algorithms on randomly generated problems using set 1 by Dobson [2].

Algorithms	SA	CS	PSO
Mean	8.273	8.273	8.273
Min.	0.951	0.951	0.951
Max.	20.905	20.905	20.905
Std. Dev.	4.979	4.979	4.979

Table 10: Comparison of relative deviation from TCL for algorithms on randomly generated problems using set 2 by Dobson [2].

Algorithms	SA	CS	PSO
Mean	7.154	7.154	7.154
Min.	0.834	0.834	0.834
Max.	22.529	22.529	22.529
Std. Dev.	4.165	4.165	4.165

Table 11: Comparison of relative deviation from TCL for algorithms on randomly generated problems using set 3 by Dobson [2].

Algorithms	SA	CS	PSO
Mean	22.308	22.308	22.308
Min.	5.347	5.347	5.347
Max.	57.436	57.436	57.436
Std. Dev.	10.995	10.995	10.995

C. Numerical Experiment 3

The third analysis is based on random data generated as shown in Table 3 from minimum and maximum value of base demand, setup cost, production rate, setup time, holding cost obtained from [1]) problem. Ten data generated for each problem size of ten, fifteen, twenty, twenty five and thirty and for each utilization level of 65%, 70%, 75%, 80%, 85%, and 90%.

Table 12, Table 13, and Table 14 obtained by solving above randomly generated problem using SA, CS and PSO respectively. CS algorithm found the same solution for high utilization cases (i.e., 85% and 90%) of all problem sizes (i.e., 10, 15, 20, 25, and 30). However, the solution found by these algorithms differs in low to medium utilization cases (i.e., 65%, 70%, 75%, and 80%). Table 12, Table 13, and Table 14 shows that 93% of solutions obtained using SA are either better or similar to best result obtained from any other algorithm, 63% of solutions obtained using CS are either better or similar to best result obtained from any other algorithm, while only 47% of solution obtained using PSO are either better or similar to best result obtained using any other algorithm.

It is important to note that in first analysis 77% of solution found by CS algorithm was better, in second analysis all algorithm did equally well, in third analysis 93% of solution found by SA are better. The deviation of result between algorithms in analysis 1 and analysis 3 is due to location of best solution in the search space. CS and PSO algorithms performed better when the best solution located far from initial feasible solution in the search space (i.e., CS search follow levy distribution while PSO search follows position and velocity to find the feasible solution) but SA performed better if the best solution is within the neighborhood of the initial feasible solution. In analysis 1 either best solutions were initial solution (i.e., for high utilization case where all k_i were '1') or far from the initial solution (i.e., for low utilization cases where k_i have different values) but in analysis 3 all best solutions are very closed to initial feasible solution (i.e., most of the k_i were either '1' and some of them were '2').

Table 12: SA algorithm's relative deviation from TCL on randomly generated problems using distribution given in Table 3.

Problem size\ Utilization		65	70	75	80	85	90
10	Mean	12.563	15.304	18.032	22.123	26.754	31.723
	Min.	1.969	1.970	2.904	6.731	11.054	12.479
	Max.	27.254	33.732	42.593	55.256	72.727	89.281
	Std. Dev.	9.8223	12.528	14.520	16.880	19.615	22.569
15	Mean	11.904	13.397	15.448	17.928	20.228	22.509
	Min.	5.092	5.661	6.152	6.542	6.839	7.142
	Max.	22.538	24.649	26.537	34.957	42.550	51.049
	Std. Dev.	5.062	5.730	6.966	8.909	10.646	12.815
20	Mean	14.043	15.746	17.573	19.356	20.990	22.284
	Min.	8.142	9.542	10.992	12.357	13.517	14.383
	Max.	18.585	20.692	23.511	26.521	30.533	33.727
	Std. Dev.	3.427	3.620	4.048	4.742	5.515	6.192
25	Mean	15.851	17.597	19.332	20.991	22.428	23.511
	Min.	11.233	12.636	13.829	14.513	15.098	15.533
	Max.	25.098	27.789	29.908	31.805	33.420	34.620
	Std. Dev.	4.444	4.831	5.088	5.356	5.632	5.871
30	Mean	19.795	21.811	23.819	25.693	27.276	28.449
	Min.	11.074	11.786	12.427	12.965	13.382	13.673
	Max.	41.068	48.401	55.387	61.848	67.291	71.323
	Std. Dev.	9.609	11.422	13.139	14.725	16.061	17.051

Table 13: CS algorithm's relative deviation from TCL on randomly generated problems using distribution given in Table 3.

Problem size\ Utilization		65	70	75	80	85	90
10	Mean	12.563	15.141	17.967	22.156	26.754	31.723
	Min.	1.969	1.970	2.904	6.731	11.054	12.479
	Max.	27.254	33.557	42.130	55.256	72.727	89.281
	Std. Dev.	9.822	12.327	14.417	16.901	19.615	22.569
15	Mean	12.321	13.916	15.836	18.052	20.228	22.509
	Min.	6.073	5.953	6.152	6.542	6.839	7.142
	Max.	22.538	25.151	27.553	35.735	42.550	51.049
	Std. Dev.	4.832	5.812	7.417	9.122	10.646	12.815
20	Mean	14.295	15.855	17.573	19.356	20.99	22.284
	Min.	8.142	9.542	10.992	12.357	13.517	14.383
	Max.	19.183	20.692	23.511	26.521	30.533	33.727
	Std. Dev.	3.707	3.614	4.048	4.742	5.515	6.192
25	Mean	16.013	17.612	19.336	20.991	22.428	23.511
	Min.	11.233	12.635	13.829	14.513	15.098	15.533
	Max.	25.985	27.901	29.908	31.805	33.420	34.620
	Std. Dev.	4.667	4.858	5.088	5.356	5.632	5.871
30	Mean	19.849	21.811	23.819	25.693	27.276	28.449
	Min.	11.074	11.786	12.427	12.965	13.382	13.673
	Max.	41.408	48.401	55.387	61.848	67.291	71.323
	Std. Dev.	9.688	11.422	13.139	14.725	16.061	17.051

Table 14: PSO algorithm's relative deviation from TCL on randomly generated problems using distribution given in Table 3.

Problem size\ Utilization		65	70	75	80	85	90
10	Mean	12.878	15.517	18.300	24.220	27.122	31.772
	Min.	1.969	1.970	5.125	7.066	11.054	12.479
	Max.	27.254	33.557	42.558	68.819	76.407	89.281
	Std. Dev.	9.632	12.031	14.202	20.343	20.584	22.577
15	Mean	12.376	14.517	16.476	18.052	20.228	22.509
	Min.	6.073	5.953	6.152	6.542	6.839	7.142
	Max.	22.538	31.157	32.113	35.735	42.550	51.049
	Std. Dev.	4.830	7.237	8.589	9.122	10.646	12.815
20	Mean	14.403	15.855	17.573	19.356	20.990	22.284
	Min.	8.142	9.542	10.992	12.357	13.517	14.383
	Max.	19.183	20.692	23.511	26.521	30.533	33.727
	Std. Dev.	3.565	3.614	4.048	4.742	5.515	6.1916
25	Mean	16.013	17.612	19.336	20.991	22.428	23.511
	Min.	11.233	12.635	13.829	14.513	15.098	15.533
	Max.	25.985	27.901	29.908	31.805	33.42	34.62
	Std. Dev.	4.667	4.858	5.088	5.3557	5.6319	5.8705
30	Mean	19.849	21.811	23.819	25.693	27.276	28.449
	Min.	11.074	11.786	12.427	12.965	13.382	13.673
	Max.	41.408	48.401	55.387	61.848	67.291	71.323
	Std. Dev.	9.688	11.422	13.139	14.725	16.061	17.051

11. Conclusion

This research presented hybridization scheme based on multiple intelligent techniques with GSS to solve the ELSP problem under basic period approach. This hybrid technique used PSO, CS and SA optimization to find the optimum value of k_i's, followed by GSS to find the basic period T. The feasibility of the solution is guaranteed with a constraint that ensures the items assigned in each period can be produced within the length of the period. The experimental results indicate following outcomes:

- The hybridization scheme was able to find comparatively better basic period solutions than GA [5] for the low utilization problems.
- The hybridization scheme was also able to find comparatively better basic period solutions than GA [5] for the high utilization problems.
- CS and PSO based hybridization algorithms performed better than SA based hybridization algorithm, if the best solution is located far from initial feasible solution in the search space.
- SA based hybridization algorithm performed better than CS and PSO based hybridization algorithms, if the best solution is much closer to the neighborhood of the initial feasible solution.

Acknowledgements:

The authors acknowledge the support of Faculty of Engineering, Sciences & Technology, Iqra University for providing research facilities and financial assistance.

Corresponding Author:
Syed Hasan Adil
Department of Computer Science
Main Campus, Iqra University
Defence View,
Shaheed-e-Millat Road (Ext.)
Karachi-75500, Pakistan
E-mail: hasan.adil@iqra.edu.pk

References

[1] Bomberger E. A dynamic programming approach to a lot size scheduling problem. Management Science 1966; 12(11): 778–84.
[2] Dobson G. The Economic Lot Scheduling Problem: Achieving Feasibility using Time-Varying Lot Sizes. Operation Research 1987; 35(5): 764-71.
[3] Bourland KE. Production planning and control for the stochastic economic lot scheduling

problem (scheduling). University of Michigan 1991.

[4] Hanssmann F. Operations Research in Production and Inventory. John Wiley and Sons 1962; 158-60.

[5] Khouja M, Michalewicz Z, Wilmot M. The use of genetic algorithms to solve the economic lot size scheduling problem. European Journal of Operational Research 1998; 110(3): 509-24.

[6] Elmaghraby SE. An Extended Basic Period Approach to the Economic Lot Scheduling Problem (ELSP). Production and Industrial Systems: Future Development and the Role of Industrial and Production Engineering, Taylor and Francis 1977: 649–62.

[7] Aytug H, Khouja M, Vergara FE. Use of genetic algorithm to solve production and operations management problems: A review. International Journal of Production Research 2003; 41(17): 3955–4009.

[8] Ben-daya M, Al-Fawzan M. A tabu search approach for the flow shop scheduling problem. European Journal of Operational Research 1998; 109(1): 88–95.

[9] Eglese RW. Simulated annealing: A tool for operational research. European Journal of Operational Research 1990; 46(3): 271–81.

[10] Jahanzaib M, Masood SA, Nadeem S, Akhtar K. A Genetic Algorithm (GA) Approach for the Formation of Manufacturing Cells in Group Technology. Life Sci J 2013; 9(4):799-809.

[11] Zanoni S, Segerstedt A, Tang O, Mazzoldi L. Multi-product economic lot scheduling problem with manufacturing and remanufacturing using a basic period policy. Computers & Industrial Engineering 2012; 62(2): 1025-33.

[12] Bulut O, Tasgetiren MF, Fadiloglu MM. A Genetic algorithm for the economic lot scheduling problem under extended basic period approach and power of two policy. Advanced Intelligent Computing Theories and Applications with Aspect of Artificial Intelligence, LNCS 2012; 6839(1): 57-65.

[13] Luo R. New algorithm for economic lot scheduling problem. International Conference on Logistics Systems and Intelligent Management 2010; 334-37.

[14] Zanoni S, Segerstedt A, Tang O, Mazzoldi L. Multi-product economic lot scheduling problem with manufacuring and remanufactuing using a basic period policy. Computers and Industrial Engineering 2012; 62(4): 1025-33.

[15] Chan HK, Chung SH, Chan TM. Combining genetic approach and integer programming to solve multi-facility economic lot scheduling problem. Journal of Intelligent Manufacturing 2012; 23(6): 2397-2407.

[16] Raza SA, Akgunduz A. A comparative study of heuristic algorithms on Economic Lot Scheduling Problem. Computer & Industrial Engineering 2008; 55(1), 94-109.

[17] Sun H, Huang H, Jaruphongsa W. A genetic algorithm for the economic lot scheduling problem under extended basic period and power-of-two policy. CIRP Journal of Manufacturing Science and Technology 2009; 2(1): 29-34.

[18] Tasgetiren MF, Bulut O, and Fadiloglu MM. A discrete harmony search algorithm for the economic lot scheduling problem with power of two policy. IEEE World Congress on Computational Intelligence 2012.

[19] Elmaghraby SE. The Economic Lot Scheduling Problem (ELSP): Review and Extensions. Management Science 1978; 24(6), 587–98.

[20] Press WH, Tehkolsky SA. Numerical Recipes 3rd Edition: The Art of Scientific Computing. Cambridge University Press, 2007.

[21] Yang XS, Deb S. Cuckoo search via L´evy flights. Proc. World Congress on Nature & Biologically Inspired Computing 2009; 210-14.

[22] Srinivasan TR, Shanmugalakshmi R. Optimizing Grid Scheduling with Particle Swarm Optimization. Life Sci J 2013; 10(4s): 559-563.

[23] Darrall H, Jacobson SH, Johnson AW. The theory and practice of simulated annealing Handbook of metaheuristics. Springer US 2003;. 287-319.

[24] Laarhoven PJ, Aarts EH. Simulated Annealing: Theory and Applications. Springer 1987.

[25] Renukadevi NT, Thangaraj P. Improvements in RBF Kernel using Evolutionary Algorithm for Support Vector Machine Classifier. Life Sci J 2013; 10(7s): 454-459.

[26] Kennedy J, Eberhard RC. Swarm intelligence. Morgan Kaufmann Publishers 2001.

[27] Gaafar L. Applying genetic algorithms to dynamic lot sizing with batch ordering. Computers & Industrial Engineering 2006; 51(3): 433–44.

6/22/2013