

SIMULATION BASED OPTIMIZATION USING PSO IN MANUFACTURING FLOW PROBLEMS: A CASE STUDY

Sai Phatak
Jayendran Venkateswaran

Industrial Engineering and Operations Research
Indian Institute of Technology, Bombay
Mumbai, INDIA

Gunjan Pandey
Shirish Sabnis
Amit Pingle

John Deere Technology Centre
John Deere India Pvt. Ltd.
Pune, INDIA

ABSTRACT

This paper presents the use of simulation based optimization in addressing manufacturing flow problems at a heavy equipments manufacturer. Optimizing the buffer allocation in an assembly line and optimizing the worker assignment at workstations are two independent problems addressed, with the objective to maximize throughput rate. The simulation models of the system, built using an in-house tool based on SLX, is interfaced with a custom designed meta-heuristic based on Particle Swarm Optimization (PSO). Two versions of the PSO have been developed: one with integer decision variables (for buffer space allocation) and another with binary variables (for worker assignment). The performance of the proposed simulation based optimization scheme is illustrated using case studies.

1 INTRODUCTION

Discrete-event simulation is popularly used in modelling and analysing complex manufacturing systems. Its use and contribution towards decision making is greatly enhanced by interfacing it with an optimum seeking package, to identify the 'best' system configuration that results in the 'optimum' performance. In such simulation-based optimization, the optimum seeking package (viz. meta-heuristic) proposes a system configuration (solution) to the simulation model. The simulation model then evaluates the given configuration and returns the corresponding performance measures. The meta-heuristic computes the next configuration for evaluation, based on the simulation output. This iterative interaction between the meta-heuristic and simulation continues until the stopping criteria is met, which can be either based on the number of iterations or time or the percentage improvement in the objective function. The meta-heuristic uses randomized but 'intelligent' schemes to search the solution space and converge to the best possible solution. Popular meta-heuristics include Scatter Search, Genetic Algorithm, Particle Swarm Optimization, Tabu Search, Response Surface Methodology and others. Now, most simulation models of manufacturing system, by themselves, are quite large and need significant time and resources (computing) to evaluate single configuration. This necessitates the meta-heuristics to *not* evaluate too many 'bad' solutions and to converge to the best solution within reasonable number of iterations. Typical manufacturing flow problems include identification of buffer and resource capacities to maximize throughput, schedule/sequence jobs at machines to minimize make span, design plant layouts and product flows to minimize takt time etc.

In the heavy equipments manufacturer under study, the assembly line operations are vital. Figure 1 depicts a typical assembly line, consisting of a main line and feeder lines. The main product assembly takes place on the main line while the feeder lines provide supplementary parts to it. The line also consists of buffers (shown by triangles) in between workstations (shown by rectangles) to facilitate the flow, avoid blocking of the assembly line and absorb the variations due to stochasticity in the processing times. At each workstation, one or more workers (shown by circles) are assigned, depending on the work content. The

assembly line handles multiple products as well as multiple variants of a product, adding to the complexity of the operations. Thus, the performance of the line will depend on the mix of products handled in the given planning horizon.

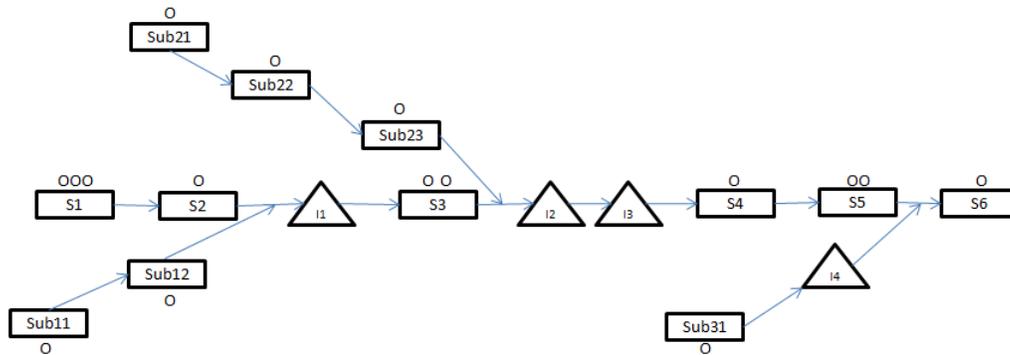


Figure 1: Typical assembly line with main line and feeder lines.

In this paper, two assembly line problems are modelled and optimized using simulation-based optimization. The first problem is the buffer or in-process kanban allocation problem. This involves the identification of the optimum number and location of buffer in the assembly line (main+feeder lines) subject to space availability and other layout constraints, to maximize throughput. Given the nature of the product (heavy equipments), buffer spaces require significant area in the shop floor which may not be always possible, and hence the need to optimize the same. The second problem is that of worker assignment. This involves the assignment of workers to specific work content at workstations, subject to utilisation requirements, travel distance and layout constraints, to maximize throughput. Traditional methods of work content allocation fared poorly in the face of dynamic demand involving multiple product variants, and in reality, it was observed that the utilisation among workers varied significantly.

The complex nature of the assembly line operations, stochastic process times, product mix, varied product flows necessitated the use of simulation model. Now, each of the above problems have a very large solution space, and hence meta-heuristic based optimum seeking package is interfaced with simulation. Particle Swarm Optimization (PSO) (Kennedy and Eberhart 1995) is the meta-heuristic used. PSO has been shown to be quite effective in many areas of application (Eberhart and Shi 2001). However, most of these applications consider continuous decision variables. In this work, buffer allocation problem involves integer decision variables and worker assignment problem involves binary decision variables. A modified PSO algorithm to handle the integer and binary variables separately, are proposed, implemented and evaluated.

Section 2 describes the assembly line problems addressed in this paper. Section 3 describes the basic PSO algorithm and Section 4 explains the two variants of PSO proposed. Section 5 throws light on the implementation and evaluation of the two variants of the algorithm, followed by the conclusions.

2 ASSEMBLY LINE PROBLEMS ADDRESSED

2.1 Buffer Space Allocation (BSA)

Buffer space is the empty space between two workstations. An ideal situation is when there are no kanbans and the products flow one at a time from one workstation to another. This is possible only if tasks at each workstation take equal time to perform. However, in reality, this is not possible due to the distribution of work content, stochastic process times, varied work content due to product mix, etc. A buffer space is required between two consecutive workstations when the process time at the downstream is more than that in the upstream workstation. This prevents blocking of the upstream workstation when the downstream workstation is busy (Pingle, Sabnis, and Pandey 2013). Now, in a mixed model line (line with many product

variants), the processing time at a workstation varies for each unit. Buffers are useful to absorb such type of variations too (Pingle, Sabnis, and Pandey 2013). The advantages of using buffers are as follows:

- Buffers control the amount of raw material and work-in-process (WIP).
- Buffers, when placed properly, result in smooth flow of products.

The objective of the buffer space allocation problem is to maximize throughput over the planning horizon, by locating minimum number of buffers on the assembly line. The decision variables are non-negative integers, indicating the number of buffers between two workstations. The input to this model is the planned production sequence over the planning horizon.

2.2 Worker Assignment (WA)

Every skilled or unskilled worker on the line has to complete the set of tasks assigned to him/her compulsorily. However, a worker may be assigned secondary tasks which can be done when the worker is idle after completing the primary tasks early. Thus, instead of the worker remaining idle he/she can help other workers on a nearby workstation, resulting in a higher utilization. Assigning a voluntary set of tasks increases the utilization of every worker. Utilization is defined as the percentage of the total time a worker does value added work. Increasing the utilization of every worker automatically increases the throughput. The utilization of every worker being high is often seen as one of the criterion for an assembly line to be performing well. The objective of the worker allocation problem is to maximize throughput over the planning horizon, by assigning secondary work content to each worker. The decision variables are binary, indicating if particular work content has been assigned as a secondary task to a worker. The input to this model is the planned production sequence over the planning horizon.

3 OVERVIEW OF PARTICLE SWARM OPTIMIZATION (PSO)

3.1 Background on PSO

Particle swarm optimization is an example of swarm intelligence metaheuristic. PSO was initially introduced in 1995 by James Kennedy and Russell Eberhart. It uses the metaphor of the flocking behaviour of birds to solve optimization problems. PSO does not use the gradient of the problem being optimized, and hence does not require the problem to be differentiable as is required by classic optimization methods such as gradient descent and quasi-newton methods. Also, PSO is not largely affected by the size and non-linearity of the problem and can converge to optimal solution in many problems where most analytical methods fail to converge (AlRashidi and El-Hawary 2009).

In the PSO algorithm the basic entity is called a particle which is like a bird in a flock. These particles are generated randomly in the search space. Each particle is a solution to the problem and has a velocity, a location in the search space and has a parameter which helps it in remembering its best position till then. The set of particles is analogous to a swarm which consists of N particles flying around in a D -dimensional search space. Also, every particle swarm has some sort of topology describing the interconnections among the particles. The set of particles to which a particle i is connected is called i 's neighbourhood. The neighbourhood maybe entire population or some subset of it. Two common topologies that are used to identify another particle to influence the individual are known as *gbest* (global best), and *lbest* (local best) or *pbest* (particle best) (Boussaid, Lepagnot, and Siarry 2013).

The positions and the velocities of all the particles are generated randomly in the initialization phase of PSO. In each iteration a particle i changes its position X_i and velocity V_i along each dimension d of the search space. This change is based on the best position P_i , it has encountered so far in its flight (also called personal best for the particle) and the best position P_g found by any other particle in its topological neighbourhood (also called global best). The velocity defines the distance, the direction and the distance the particle should fly (Boussaid, Lepagnot, and Siarry 2013).

In order to avoid the particles from flowing out of the allowed search space, Kennedy and Eberhart defined a clamping scheme to limit the velocity of each particle, so that each component V_i is kept within

the range $[-V_{max}, V_{max}]$ (Kennedy and Eberhart 1995). Since the choice of V_{max} affects the balance between exploration and exploitation, the value of V_{max} needs to be carefully chosen (Boussaid, Lepagnot, and Siarry 2013).

To overcome the premature convergence of the PSO, which was a major drawback of the original PSO, many strategies have been developed. However, the most popular are inertia and constriction. The inertia weight ω , plays the role of balancing the global search and local search. It can be a positive constant or even a positive linear or non-linear function of time. A large inertia weight encourages global exploration (i.e., diversifies the search in the whole search space) while a smaller inertia weight encourages local exploitation (i.e., intensifies the search in the current region). Also there are two acceleration constants c_1 and c_2 . They are constant multiplier terms known as acceleration coefficients. c_1 is the cognitive coefficient while c_2 is the social coefficient. They represent the attraction that a particle either has towards its own success (the cognitive part) or towards the success of its neighbours (the social part) respectively (Boussaid, Lepagnot, and Siarry 2013). The PSO algorithm has proved to be effective in many areas of applications (Eberhart and Shi 2001). It has been used to evolve artificial neural networks which in turn have been applied to the analysis of human tremor. It has been used for reactive power and voltage control by a Japanese electrical utility (Eberhart and Shi 2001). The algorithm has also been used for the optimization of electric power systems (AlRashidi and El-Hawary 2009). However most of these applications optimize a continuous objective function. Very few areas have been explored with PSO as far as discrete objective functions are concerned.

4 PROPOSED PSO ALGORITHMS

4.1 PSO for Integer Decision Variables: Buffer Space Allocation

In case of integer decision variables (for buffer space allocation), the position vector is n dimensional vector where each element takes only integer values. Also, there can be a separate matrix for denoting the maximum value each dimension d of the position matrix can take. There can be a limitation on the summation of the position vector. In this variant of PSO the velocity is updated according to the following equation (Kennedy and Eberhart 1995):

$$V_{id}[t + 1] = \omega * V_{id}[t] + c_1 * r_1 * (P_{id}[t] - X_{id}[t]) + c_2 * r_2 * (G_d[t] - X_{id}[t]) \quad (1)$$

where

- V_{id} : Velocity of dimension d of particle i at iteration t
- ω : Weighting Function/inertia weight
- c_j : Weighting factor (c_1 is cognitive constant and c_2 is social constant)
- V_{id} : Velocity of dimension d of particle i at iteration t
- r_1 : random number between 0 and 1
- r_2 : random number between 0 and 1
- X_{id} : Current position of dimension d of particle i at iteration t
- P_{id} : Local best position so far of dimension d of particle i
- G_d : Best position of dimension d of the all the particles in the swarm
- N : number of particles

The inertial weight ω is determined by $\omega = 0.9 - (niter * 0.5/totaliter)$ where

- $niter$: current iteration
- $totaliter$: total iterations

Equation (1) specifies that the velocity of a particle at iteration t is determined by the previous velocity of the particle, the cognition part, and the social part. The values $c_k * r_k$ ($k=1, 2$) determine the weights of the two parts, where their sum is usually limited to V_{max} . The position of each particle is also updated in each iteration by adding the velocity vector to the position vector. The position is updated according to

the following equation (Kennedy and Eberhart 1995):

$$X_{id}[t + 1] = V_{id}[t + 1] + X_{id}[t] \quad (2)$$

The general structure of the proposed PSO algorithm for handling integer variables is as follows:

- Step 1:** Set the parameters N , c_1 , c_2 , $\omega = 0.9 - (niter * 0.5/totaliter)$, V_{max} , $V_i=0$ of PSO.
- Step 2:** Initialize a population of N particles with random positions, $X_{id}[1]$ and velocities, $V_{id}[1]$ in the problem space.
- Step 3:** Evaluate the desired optimization fitness function $f()$ for each particle using simulation. Set $P_{id}[1] = X_{id}[1]$, $G_d[1] = X_{id}[1]$.
- Step 4:** For each Individual particle, compare the particle's fitness value with its $pbest$ ($f(P_{id}[t])$). If the current value ($f(X_{id}[t])$) is better than the $pbest$ ($f(P_{id}[t])$) value, then set $X_{id}[t]$ as the $pbest$ ($P_{id}[t]$) for particle i .
- Step 5:** Identify the particle that has the best fitness value. The value of its fitness function is identified as $gbest$ ($f(G_d[t])$).
- Step 6:** Compute the new velocities ($V_{id}[t + 1]$) and positions ($P_{id}[t + 1]$) for each particle according to equations (1) and (2) respectively.
- Step 7:** Repeat steps 3-6 until the stopping criterion of maximum generations is met.

This version of PSO algorithm has been implemented for optimizing the buffer space in the assembly line. The position vector indicates the buffer capacity of the corresponding buffer. For example, $X[10] = 4$ implies that at buffer position 10 a buffer of capacity 4 is located. There is an upper bound on the maximum buffer capacity (size) at each buffer position. Also, there is a limitation on the total number of buffers that can be placed in the entire assembly line. However in the buffer space optimization problem, only the main line buffers have been considered for optimization. The buffers from feeder lines have not been considered.

4.2 PSO for Binary Decision Variables: Worker Assignment

In case of binary decision variables (for worker assignment), the position matrix is a $d1 \times d2$ matrix, where each element takes binary values (0 or 1). If a particular dimension of the row (worker) is associated with a particular dimension of the column (task) then the corresponding matrix element is 1 and 0 otherwise.

In this variant of PSO, velocity is computed as follows:

$$V_{id1d2}[t + 1] = \omega * V_{id1d2}[t] + c_1 * r_1 * (P_{id1d2}[t] - X_{id1d2}[t]) + c_2 * r_2 * (G_{d1d2}[t] - X_{id1d2}[t]) \quad (3)$$

where

- V_{id1d2} : Velocity of dimension $d1$ and $d2$ of particle i at iteration t
- ω : Weighting Function/inertia weight
- c_j : Weighting factor (c_1 is cognitive constant and c_2 is social constant)
- V_{id1d2} : Velocity of dimension $d1$ and $d2$ of particle i at iteration t
- r_1 : random number between 0 and 1
- r_2 : random number between 0 and 1
- X_{id1d2} : Current position of dimension $d1$ and $d2$ of particle i at iteration t
- P_{id1d2} : Local best position so far of dimension d of particle i
- G_{d1d2} : best position of dimension $d1$ and $d2$ of the all the particles in the swarm
- N : number of particles

The inertial weight ω is determined by $\omega = 0.9 - (niter * 0.5/totaliter)$ where

niter : current iteration
totaliter : total iterations

Equation (3) specifies that the velocity of a particle at iteration t is determined by the previous velocity of the particle, the cognition part, and the social part. The values $c_k * r_k$ ($k=1, 2$) determine the weights of the two parts, where their sum is usually limited to V_{max} . By (3), each particle moves according to its new velocity. In this version the particles are represented by binary variables. Hence the velocity trail values need to be converted from real numbers to the changes of probabilities (Kennedy and Eberhart 1997). For the velocity value of each bit in a particle, Kennedy and Eberhart claim that higher value is more likely to choose 1, while lower value favours 0. The velocity value is constrained to the interval $[0, 1]$ by using the following sigmoid function (Kennedy and Eberhart 1997):

$$sig(V_{id1d2}[t]) = 1/(1 + exp(-V_{id1d2}[t])) \quad (4)$$

To avoid $sig(V_{id1d2}[t])$ approaching 0 or 1, a constant V_{max} is used to limit the range of $V_{id1d2}[t]$. In practice, V_{max} is often set to 4, i.e., $V_{id1d2}[t] \in [-V_{max}, V_{max}]$.

Now $sig(V_{id1d2}[t])$ represents the probability of $V_{id1d2}[t]$ taking the value 1. For example, $sig(V_{id1d2}[t])=0.2$ in represents that there is a 20% chance that the position matrix element X_{d1d2} will be 1 (Liao, Tseng, and Luarn 2007).

4.2.1 Constructing a Particle Sequence

Each particle constructs its new sequence based on its changes of probabilities from the velocity trail. However, the position is not directly determined on the value of the sigmoid function. The values of the sigmoid function are summed up. It can be represented as follows (Liao, Tseng, and Luarn 2007):

$$m_{id1} = \sum_{d2 \in F} sig_{id1d2} \quad (5)$$

where F is the set of all stages on which the worker is allowed to work on the assembly line. This value is calculated for all the rows. Now a random number is generated for each element in the matrix which is compared with $(sig_{id1d2})/m_{id1}$. If this random number is greater than the compared value then the corresponding position i.e. X_{id1d2} is assigned 1. The general structure of the changed PSO algorithm is as follows:

- Step 1:** Set the parameters $N, c_1, c_2, \omega = 0.9 - (niter * 0.5/totaliter), V_{max}, V_i=0$ of PSO.
- Step 2:** Initialize a population of N particles with random positions ($X_{id1d2}[1]$) and velocities ($V_{id1d2}[1]$) in the problem space.
- Step 3:** Evaluate the desired optimization fitness function ($f()$) for each particle and set $P_{id1d2}[1] = X_{id1d2}[1], G_{d1d2}[1] = X_{id1d2}[1]$.
- Step 4:** For each individual particle, compare the particles fitness value with its pbest ($f(P_{id1d2}[t])$). If $f(X_{id1d2}[t])$ is better than the pbest ($f(X_{id1d2}[t])$), then set $X_{id1d2}[t]$ as the pbest ($P_{id1d2}[t]$) for particle i .
- Step 5:** Identify the particle that has the best fitness value. The value of its fitness function is identified as gbest ($G_{d1d2}[t]$).
- Step 6:** Compute the new velocities according to equation (3).
- Step 7:** Convert the velocities to sigmoid function ($sig(V_{id1d2}[t])$).
- Step 8:** Sum up all the sigmoid function values of the matrix elements on which the worker is allowed to work on the assembly line.i.e. calculate m_{id1} .
- Step 9:** Generate a random number for every element of the matrix and compare it with $(sig_{id1d2})/m_{id1}$ where if the random number is greater then assign 1 to the corresponding position.
- Step 10:** Repeat steps 8 and 9 till all positions are covered for all particles.
- Step 11:** Repeat steps 3-10 until the stopping criterion of maximum generations is met.

This version of PSO algorithm has been implemented for optimizing the worker assignment on an assembly line. The rows of the position matrix represent the workers while the columns represent the workstations. If a worker works on a particular workstation then the corresponding matrix element is 1 else 0. However the primary workstation of every worker always remains 1. A matrix with workers and their corresponding primary workstations is provided as an input.

5 EXPERIMENTAL RESULTS

The simulation models of the assembly lines (testcases) have been built in using an in-house tool based on the simulation package SLX (Henriksen 1996). The complexity of the testcases ranges from medium to high in terms of number of stations (main and sub assemblies) and workers. The algorithms (both variants) have been separately implemented and interfaced with simulation using SLX.

The optimization algorithm PSO has been written in SLX. However, it is interfaced with the in-house developed toolset since SLX doesn't have graphical user interface. Simulation analysis is performed using the in-house developed toolset. The basic structure of this simulation toolset consists of: Microsoft Access Database to store the user input and to write outputs in predefined tables; Graphical User Interface (GUI) is developed using Microsoft Visual Studio and the simulation engine is developed in SLX (Simulation Language with eXtendability) from Wolverine Software CorporationTM (Henriksen 2011). When the simulation toolset is launched for the first time, it establishes a relationship between the GUI and the database tables. During the simulation run the simulation engine reads the data from the database tables and executes the input model (Pingle, Sabnis, and Pandey 2013). At the end of one simulation run, the results are given to the algorithm. The algorithm then evaluates new positions, velocities, global best and individual best of each of the particles. A simulation run based on the new values is again carried out. This continues for the given number of iterations and the global best value obtained is displayed. The flow of the program is described in Figure 2.

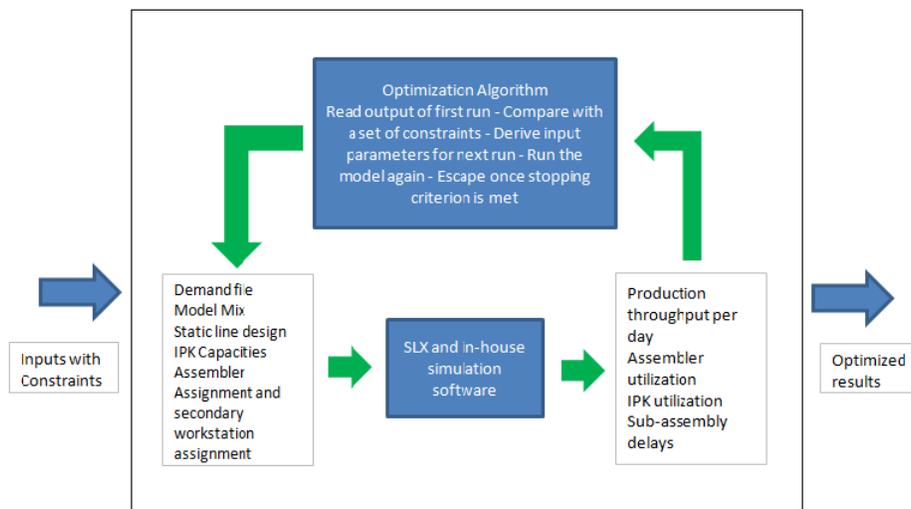


Figure 2: Flow of program.

The system hardware configurations used are a 64 bit with Intel Core i5 processor with 48 GB RAM. The parameters of PSO used for these test functions (for both variants) is as follows: Number of particles = 10; Number of iterations = 50; $c_1 = 2$; and $c_2 = 2$.

In all the test scenarios described, the demand file (demand for the entire simulation period) was read by SLX before the simulation run began. Raw material is always available before the 1st workstation and also infinite sink for the accommodation of finished products is assumed to be present at the end of

the assembly line. Though work content variation is considered in the assembly line, there is not much stochasticity in the process times of work stations. This eliminates the need of multiple runs for one combination. However, machine breakdowns are not considered in this simulation. In BSA only the main line buffer spaces have been considered for optimization. In every assembly line, a specific set of work is assigned to each worker. With worker allocation, a different combination of work is assigned to every worker so that the average utilization improves along with throughput. Because of the nature of products, high optionality (different variants of the same product) is observed resulting in varied work content. In addition to this, different products assembled on the same assembly line also results in varied work content. In each of the testcases, the work content (total time a product takes on the assembly line for completion), description of buffers, sub-assemblies, worker assignments, product mix is mentioned. This gives a basic idea about the assembly line.

In BSA the demand file has been generated randomly for the given simulation period. However, for WA a predefined sequence is determined using a scheduling algorithm and that has been used as the demand file for WA. This demand file sequence has been used for running WA base case.

5.1 Buffer Space Allocation

Description of Testcase 1. The test case consists of an assembly line with 13 workstations on the main line, 1 subassembly with 1 workstation on it, 8 workers, with a planning horizon of 5 days. There is no warm up period for this test scenario. The target throughput is 200 per day. The critical point for buffers is between the 6th and 7th workstation, because the base model consists of 20 buffer spaces in between these two workstations. Also, the capacity of buffer size between 11th and 12th workstation is 4. For all the remaining stations the buffer capacity is either 1 or 2. The total number of buffers in the base model is 39. 1 subassembly merges with the main line at station 12.

In the base model, number of parts produced are 811. This is the last complete part that comes out of the assembly line on the last day (Day 5) at 11:48 pm. The work content variation varies in the range of 43.18 to 81.11 minutes. The variation in finished products is none. Thus, the end product of this assembly line is only 1 finished product. The cycle time on the workstation varies from 0.02 minutes to 1.5 minutes.

After optimizing the assembly line for buffer space, number of parts produced in 5 days are 880. This is the last complete part that came out of the assembly line on the last day (Day 5) at 11:35 pm. The optimum result is obtained in iteration 4 for 6th particle. Hence the time required to actually reach the optimum is around 9 minutes. Though the target throughput is not met, the average output as well as the average worker utilization increased. Also, two buffer spaces are saved. The details of the base model and optimized model are mentioned in Table 1.

Table 1: BSA Real time Testcase 1 results.

Base Model output			
	Average	Minimum	Maximum
Worker utilization	60.34	28.54	95.63
Average output(unit) per day	162.2	137	197
Number of buffers	39		
Buffer Optimization output			
	Average	Minimum	Maximum
Worker utilization	62.8	31.76	99.8
Average output(unit) per day	176	169	180
No of buffers	37		
Optimization run time	90 mins		

Description of Testcase 2. The test case consists of an assembly line with 6 workstations on the main line, 2 parallel subassemblies each with 1 workstation merging the main line at 1st workstation, 7 workers,

with a planning horizon of 20 days. The warm up period for this test scenario is 1 day. The target throughput is 16 per day. The buffer capacities vary from 2 to 8 with the highest buffer capacity between 4th and 5th station. The buffer capacity between 1st and 2nd station is 6. The total number of buffers in the base model is 22.

In the base model number of parts produced are 237. This is the last complete part that comes out of the assembly line on the last day (Day 20) at 1:56 pm. The work content variation varies in the range of 215.5786 to 247.76 minutes. The number of products produced by this assembly line is 23 with the product percentages varying from 0.03% to 8%. The cycle time on the workstation varies from 0.07 minutes to 2.5 minutes.

After optimizing the assembly line for buffer space, number of parts produced in 5 days are 242. This is the last complete part that came out of the assembly line on the last day (Day 20) at 2:08 pm. Also, fifteen buffer spaces are saved. The optimum results are obtained in iteration 28 for 8th particle . Hence, the time required to reach the optimum is 4 minutes. The details of the base model and optimized model are mentioned in Table 2.

Table 2: BSA Real-time Testcase 2 results.

Base Model output			
	Average	Minimum	Maximum
Worker utilization	88.3	63.51	100
Average output(unit) per day	15.8	0	19
Number of buffers	22		
Buffer Optimization output			
	Average	Minimum	Maximum
Worker utilization	89.6	64.25	100
Average output(unit) per day	16.13	0	20
No of buffers	7		
Optimization run time	7 mins		

5.2 Worker Assignment

Description of Testcase 1. The test scenario is the same as the 1st test scenario of BSA. Hence, the model description is the same as described above. However, for WA the base case results are obtained by using a predetermined sequence. Hence the base case results for WA are as follows.

The number of parts produced is 875. This is the last part that comes out of the assembly line on the last day (Day 5) at 11:24 pm.

After optimizing the assembly line for worker allocation, the number of parts produced in 5 days is 1047. This is the last complete part that comes out of the assembly line on the last day (Day 5) at 11:48 pm. Thus, after applying WA the target is reached. The optimum result for WA is obtained in iteration 335 for 5th particle. Hence the time required to actually reach the optimum is around 60 minutes. The details of the base model and optimized model are mentioned in Table 3.

Table 3: WA Testcase 1 results.

Base Model output			
	Average	Minimum	Maximum
Worker utilization	62.47	31.31	99.76
Average output per day (units per day)	175	168	179
Worker Allocation output			
	Average	Minimum	Maximum
Worker utilization	74.32	99.57	35.62
Average output per day (units per day)	209.4	197	221
Optimization run time	90 mins		

Description of Testcase 3. The test case consists of an assembly line with 17 workstations on the main line, 50 subassemblies with 1 workstation in each, 39 workers, with a planning horizon of 15 days. There is no warm up period for this test scenario. The target throughput is 42 per day. The main line does not have any buffer spaces. However, the subassemblies contain buffers with capacities ranging from 2 to 100. On an average 3-4 subassemblies merge at every workstation with the main line.

In the base model number of parts produced are 631. This is the last complete part that comes out of the assembly line on the last day (Day 15) at 11:24 pm. The work content variation varies in the range of 700.7 to 1025.42 minutes. The variation in finished products is none. Thus, the end product of this assembly line is only 1 finished product. The cycle time on the workstations vary from 0.18 minutes to 12 minutes.

After optimizing the assembly line for worker allocation, the number of parts produced in 5 days is 1047. This is the last complete part that comes out of the assembly line on the last day (Day 5) at 11:48 pm. The optimum results are obtained in iteration 19 for 6th particle. Hence the time required to reach the optimum is 11 minutes. The details of the base model and optimized model are mentioned in Table 4.

Table 4: DSO and WA real-time Testcase 3 results.

Base Model Output			
	Average	Minimum	Maximum
Worker utilization	58.21	15.9	75.69
Average output per day (units per day)	42.07	33	44
Worker Allocation Output			
	Average	Minimum	Maximum
Worker utilization	83.18	53.11	94.57
Average output per day (units per day)	60.73	65	52
Optimization run time	27 mins		

6 CONCLUSIONS

The PSO algorithm has been developed in two versions, both having a discrete nature. The first version is a 1 dimensional PSO algorithm with position taking integral values while the second version is a 2 dimensional PSO algorithm with position taking only binary values. The two versions of the algorithm developed perform satisfactorily and reaching the optimum values (best solution) for the scenarios tested. Work is in progress to evaluate and benchmark the performance of the algorithms for larger and complex assembly lines. Also, while testing for buffer allocation optimization, only the main line buffers have been considered by PSO. Subassembly and feeder line buffers are to be included for optimization in the future. Also, in worker allocation, the number of workers was fixed while only the maximization of utilization

and throughput are addressed. An additional objective of minimizing the number of workers can also be considered. Further, a decision making framework is being developed, with simulation-based optimisation as its core, to optimize all aspects of the assembly line.

ACKNOWLEDGMENTS

We would like to thank John Deere India Pvt. Ltd. for providing us the opportunity to carry out this work.

REFERENCES

- AlRashidi, M. R., and El-Hawary. 2009. "A Survey of Particle Swarm Optimization Applications in Electric Power Systems". *IEEE Transactions on Evolutionary Computation* 13 (4): 913–918.
- Boussaid, I., J. Lepagnot, and P. Siarry. 2013. "A survey on optimization metaheuristics". *Information Sciences* 237:82–117.
- Eberhart, R. C., and Y. Shi. 2001. "Particle Swarm Optimization, Development, Applications and Resources". In *Proceedings of the 2001 Congress on Evolutionary Computation*, Volume 1, 81–86. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.
- Henriksen, J. O. 1996. "An Introduction to SLX". In *Proceedings of the Winter Simulation Conference*, edited by J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain, 468–475. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.
- Kennedy, J., and R. Eberhart. 1995. "Particle Swarm Optimization". In *Proceedings of the 1995 IEEE International Conference on Neural Networks*, Volume 4, 1942–1948. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.
- Kennedy, J., and R. Eberhart. 1997. "A discrete binary version of the particle swarm algorithm". In *Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics*, Volume 5, 4104–4108. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.
- Liao, C.-J., C.-T. Tseng, and P. Luarn. 2007. "A discrete version of Particle Swarm Optimization for Flowshop Scheduling problems". *Computers and Operations Research* 34 (10): 3099–3111.
- Pingle, A., S. Sabnis, and G. Pandey. 2013. "In process kanban optimization for a manufacturing simulation". Technical report, SAE Technical Paper 2013-01-0065.

AUTHOR BIOGRAPHIES

SAI PHATAK is a Master's student in Industrial Engineering and Operations Research at Indian Institute of Technology Bombay, Mumbai, India. She did her Bachelors in Electronics and Telecommunication from V.E.S.I.T., Mumbai, India. Her research interests are in simulation modelling with applications in manufacturing systems. Her email address is saig21090@gmail.com.

JAYENDRAN VENKATESWARAN is an Assistant Professor of Industrial Engineering and Operations Research at Indian Institute of Technology Bombay, Mumbai, India. He holds M.S. and Ph.D. degrees in Systems and Industrial Engineering from the University of Arizona, Tuscon, U.S.A. His research and teaching interests are primarily in the areas of hybrid modeling, simulation and analysis of complex stochastic systems (viz. supply chains). His email address is jayendran@iitb.ac.in.

GUNJAN PANDEY is working with John Deere India Pvt. Ltd. She holds M.Tech. degree in CAD/CAM from N.I.T. Allahabad, India. Her email address is pandeygunjan@johndeere.com.

SHIRISH SABNIS is working with John Deere India Pvt. Ltd. He holds a master's degree in Mechanical Engineering from I.I.Sc. Bangalore, India. His email address is SabnisShirish@johndeere.com.

AMIT PINGLE is working with John Deere India Pvt. Ltd. He holds a master's degree in optimization from Wichita State University, U.S.A. His email address is pingleamitb@johndeere.com.