Machine Loading in Flexible Manufacturing System: A Swarm Optimization Approach

Sandhyarani Biswas
National Institute of Technology
sandhya_biswas@yahoo.co.in
S.S.Mahapatra
National Institute of Technology
mahapatrass2003@yahoo.com

Loading is one of the vital issues in flexible manufacturing system (FMS) production planning which deals with assignment of the necessary operations and tools among various machines in an optimal manner. Such a problem is combinatorial in nature and found to be NP-complete. In this paper, an attempt has been made to address such problems using mutation in particle swarm optimization (PSO) to avoid premature convergence with the objective of minimization of system unbalance. Promising results have been obtained when the solution is compared with existing techniques for ten standard problems available in literature representing three different FMS scenarios.

Keywords: Flexible Manufacturing System,Machine Loading Problem, Particle Swarm Optimization, Mutation; System Unbalance

1. Introduction

Today's dynamic production environment is characterized by a large volume of uncertainty such as rapid market changes, increased product variety, competitive prices and short product life cycles. Therefore, it is of prime importance to introduce flexible manufacturing systems (FMSs) so that these uncertainties can be handled in an effective manner. FMS is characterized as an integrated, computer-controlled complex arrangement of automated material handling devices and computer numerically controlled (CNC) machine tools that can simultaneously process medium-sized volumes of a variety of part types (Stecke, 1983). The highly integrated FMS offers the opportunity to combine the efficiency of transfer line and the flexibility of a job shop to best suit the batch production of mid volume and mid variety of products. However, flexibility has a cost, and the capital investment sustained by firms to acquire such systems is generally very high. Therefore, particular attention must be paid to proper planning of FMS during its development phase in order to evaluate the performance of the system and justify the investment incurred. Prior to production, careful operational planning is essential to establish how well the system interacts with the operations over time. Hence, successful operation of FMS requires more intense planning as compared to any conventional production system. The decisions related to FMS operations can be broadly divided into pre-release and post-release decisions. Pre-release decisions include the FMS operational planning problem that deals with the pre-arrangement of jobs and tools before the processing begins whereas post-release decisions deal with the scheduling problems. Pre-release decisions viz., machine grouping, part type selection, production ratio determination, resource allocation and loading problems must be solved while setting up of a FMS. Amongst pre-release decisions, machine loading is considered as one of the most vital production planning problem because performance of FMS largely depends on it. Loading problem, in particular, deals with allocation of jobs to various machines under technological constraints with the objective of meeting certain performance measures. Therefore, the problem is combinatorial in nature and happens to be NP-hard. Formulations of loading problems in FMS and solution techniques have drawn the attention of researchers for quite some time. FMS planning problem formulated as non-linear 0-1 mixed-integer programming (MIP) (Stecke, 1983) and subsequently a branch-and-bound algorithm was developed (Berrada and Stecke, 1986). Although analytical and mathematical programming-based methods are robust in applications yet they tend to become impractical when problem size increases. This motivated the researchers to develop fast and effective heuristics for solving loading problems in large-sized FMSs. One of the important heuristics uses the concept of essentiality ratio for maximization of throughput and minimization of system unbalance simultaneously (Mukhopadhyay et al., 1992). Later on, heuristics have been developed using fixed pre-determined job ordering rules as input while solving loading problems (Tiwari et al., (1997). However, it has been established that shortest processing time (SPT) rule works well in comparison to other rules viz., longest processing time (LPT), first-in-first out (FIFO), and last-in-first-out (LIFO) (Moreno and Ding, 1993**).** Then, multi-stage programming approach has been incorporated while developing heuristics for minimizing system unbalance (Nagarjuna et. al., 2006). The major limitation of heuristics lies in the fact that their inability
to estimate the results in a new or completely changed environment as they are generally rule-based and mostly rely on empirical data. Therefore, numerous researchers have used meta-heuristic approaches for solving the

machine loading problem. Genetic algorithm (GA) based approaches for loading problem is found to ensure optimal solution with less computational effort (Tiwari et. al., 2007). Many researchers (Kumar and Shanker, 2000 and Swarnkar and Tiwari, 2004) have addressed machine-loading problem having the bi-criterion objectives of minimizing system unbalance and maximizing the throughput using a hybrid algorithm based on tabu search (TS) and simulated annealing (SA). The main advantage of this approach is that a short-term memory provided by the tabu list can be used to avoid revisiting the solution while preserving the stochastic nature of the SA method.

Numerous methods based on mathematical, heuristics, and meta-heuristics have been suggested by the researchers in the pursuit of obtaining quality solutions to loading problems and reduce computational burden. But these approaches hardly capable of producing optimal/near optimal solutions or requires excessive computational efforts to arrive at quality solutions. In order to alleviate these difficulties, an attempt has been made in this paper to propose an algorithm based on particle swarm optimization (PSO) to solve the machine loading problem of a random FMS with the objective of minimization of system unbalance while satisfying the constraints related to available machining time and tool slots. However, PSO has inherent drawback of trapping at local optimum due to appreciable reduction in velocity values as iteration proceeds and hence reduce solution variety. This drawback has been addressed effectively by incorporating mutation, a commonly used operator in genetic algorithm, to improve the solution quality.

The remainder of this paper is organized as follows. Section 2 formally defines the problem studied in this paper along with the objectives and assumptions made to solve the problem. The proposed algorithm based on PSO is presented in section 3. In section 4, results of benchmark problems from open literature are compared with proposed method to illustrate its advantage over other methods. Finally, conclusions drawn from this study are summarized and direction for future research is outlined in section 5.

2. Problem Description

The loading problem in manufacturing deals with selecting a subset of jobs from a set of all the jobs to be manufactured and assigning their operations to the relevant machines in a given planning horizon with the technological constraints in order to meet certain performance measures such as minimization of system unbalance and maximization of throughput. System unbalance can be defined as the sum of unutilized or overutilized times on all the machines available in the system whereas throughput refers to the summation of the batch size of the jobs that are to be produced during a panning horizon. Minimization of system unbalance is equivalent to maximization of machine utilization. The processing time and tool slots required for each operation of the job and its batch size are known before hand. There are two types of operations; essential and optional associated with the part types. Essential operations can be carried out on a particular machine using a certain number of tool slots while the optional operation can be performed on a number of machines with same or different processing time and tool slots. The FMS under consideration derives its flexibility in selection of machine for optional operation of the job. Generally, the complexity of these problems depends on whether the FMS is of a dedicated type or a random type. A dedicated FMS is designed to produce a rather small family of similar parts with a known and limited variety of processing requirements while in a random-type system a large family of parts having a wide range of characteristics with random elements is produced and the product mix is not completely defined at the time of installing the system. This paper addresses the loading problem in a random FMS. The proposed approach has been tested on problems pertaining to three sizes of FMSs (the details are given in Table 1). The details of data related to problem 1 of FMS type 1 (jobs, batch size, unit processing time, machine options, number of tool slots etc.) having four machines are given in Table 2.

Table 1 Details of different FMS scenarios

FMS type	**Number of machines**	**Available time on each machines**	**Number of tool slots on each machines**
FMS 1	4	480 min, 480 min, 480 min, 480 min	5, 5, 5, 5
FMS 2	5	960 min, 960 min,960 min, 960 min,960 min	10, 12, 10, 12, 10
FMS 3	6	960 min, 960 min,960 min, 960 min,960 min, 960min	14, 14, 14, 14, 14,16

In order to minimize the complexities in analyzing the problem for a practical FMS, the mathematical model is based on the following assumptions:

- Initially, all the jobs and machines are simultaneously available.
- Processing time required to complete an entire job order is known a priori.
  Job undertaken for processing is to be completed for all its operation before considering a new job.

Table 2 Detail description of jobs of problem number 1 (FMS 1)

Job Number	Batch size	Operation number	Machine number	Unit processing time(min)	Tool slots needed	Total processing time
1	10	1	4	16	1	160
		2	4,2,3	7,7,7	1,1,1	70
2	13	1	12,3	25	1	325
		2	2,1	17	1	221
		3	1	24	1	312
3	14	1	4,1	26,26	2,2	364
		2	3	11	3	154
4	7	1	3	24	1	168
		2	4	19	1	133
5	9	1	1,4	25	1	255
		2	4	25	1	255
		3	2	22	1	198
6	8	1	3	20	1	160
7	9	1	2,3	22,22	2,2	198
		2	2	25	1	225

- This is called non-splitting of the job.
- Operation of a job once started on a machine is continued until it is completed.
- Transportation time required to move a job between machines is negligible.
- Sharing and duplication of tool slots is not allowed.

Different researchers have calculated system unbalance in different ways. Some researchers such as solved the machine loading problem by considering overloading of the machines (Mukhopadhyay et al. 1992, Nagarjuna et al., 2006 and Tiwari et. al., 2007) while others have not permitted overloading (Shanker et al., 1989). In order to examine the efficiency of the proposed algorithm, the problem described above is formulated in this paper by considering two cases. In first case, an optimal solution is obtained without permitting overloading on machines whereas overloading on machines is permitted in the second case. Mathematical formulation of the problem for both the cases are described in the followings.

2.1 Mathematical formulation

2.1.1. Notations

In order to formulate machine loading problem of FMS, the following notations are introduced:

j: job index , j = 1,2………J

m: machine index, m = 1,2,……..M

S_m: tool slot capacity of machine m

o: number of operations for job j, o = 1,2……..O_j

B_j :Batch size of job j

T_m : Length of scheduling period for m^{th} machine

P_{jom} : Processing time of operation o of job j on machine m

S_{jom}: Number of tool slots required for processing operation o of job i on machine m

B(j,o): Set of machines on which operation o of job j can be performed

SU: System Unbalance

TH: Throughput

$$X_j = \begin{cases} 1, & \text{if job } j \text{ is selected} \\ 0, & \text{otherwise} \end{cases}$$

$$X_{jom} = \begin{cases} 1, & \text{if operation } o \text{ of job } j \text{ is assigned on machine } m \\ 0, & \text{otherwise} \end{cases}$$

Case1: Overloading not permitted

Objective of the FMS loading problem is to minimize total system unbalance and is represented by Eq. (1)
Subjected to the following constraints:

$$\text{Minimize } \sum_{m=1}^{M} T_m - \sum_{m=1}^{M} \sum_{j=1}^{J} \sum_{o=1}^{O_j} B_j p_{jom} X_{jom} \qquad \text{……………………………(1)}$$

$$\sum_{j=1}^{J}\sum_{o=1}^{O_j} B_j p_{jom} X_{jom} \le T_m \qquad m = 1,2 \ldots\ldots M \qquad \ldots\ldots(2)$$

$$\sum_{j=1}^{J}\sum_{o=1}^{O_j} S_{jom} X_{jom} \le S_m \qquad m = 1,2 \ldots\ldots M \qquad \ldots\ldots(3)$$

$$\sum_{G \in B(j,o)} X_{joG} \le 1 \qquad j = 1,2 \ldots\ldots J \qquad \ldots\ldots(4)$$

$$o = 1,2 \ldots\ldots O_j$$

$$\sum_{o=1}^{O_j}\sum_{m=1}^{M} X_{jom} = X_j O_j \qquad j = 1,2 \ldots\ldots J \qquad \ldots\ldots(5)$$

The constraint, Eq. (2), ensures that overloading of machines is not permitted. Eq. (3) is to ensure the jobs will be loaded only when there is availability of tool slots on each machine. Eq. (4) ensures that a particular operation of a job is done only on one machine and Eq. (5) ensures that the job cannot be split.

Case2: Overloading permitted

Objective of the FMS loading problem is to minimize the absolute value of total system unbalance and is represented in Eq. (6).

$$\text{Minimize} \sum_{m=1}^{M}\left| T_m - \sum_{j=1}^{J}\sum_{o=1}^{O_j} B_j p_{jom} X_{jom} \right| \qquad \ldots\ldots(6)$$

Subjected to following constraints:

$$\sum_{m=1}^{M}\sum_{j=1}^{J}\sum_{o=1}^{O_j} B_j p_{jom} X_{jom} \le \sum_{m=1}^{M} T_m \qquad m=1,2 \ldots\ldots M \qquad \ldots\ldots(7)$$

$$\sum_{j=1}^{J}\sum_{o=1}^{O_j} S_{jom} X_{jom} \le S_m \qquad m = 1,2 \ldots\ldots M \qquad \ldots\ldots(8)$$

$$\sum_{G \in B(j,o)} X_{joG} \le 1 \qquad j = 1,2 \ldots\ldots J \qquad \ldots\ldots(9)$$

$$o = 1,2 \ldots\ldots O_j$$

$$\sum_{o=1}^{O_j}\sum_{m=1}^{M} X_{jom} = X_j O_j \qquad j = 1,2 \ldots\ldots J \qquad \ldots\ldots(10)$$

The constraint, Eq. (7), ensures that overloading of machines is permitted. Eq. (8) is to ensure the jobs will be loaded only when there is availability of tool slots on each machine. Eq. (9) ensures that a particular operation of a job is done only on one machine and Eq. (10) ensures that the job cannot be split.

3. Proposed Methodology

3.1. Particle Swarm Optimization

Particle Swarm Optimization (PSO) algorithm, originally introduced by Kennedy and Eberhart in 1995, is a population-based evolutionary computation technique. It is motivated by the behavior of organisms such as bird flocking and fish schooling. In PSO, each member is called particle, and each particle moves around in the multidimensional search space with a velocity which is constantly updated by the particle's own experience and the experience of the particle's neighbors or the experience of the whole swarm. The members of the entire population are maintained throughout the search procedure so that information is socially shared among individuals to direct the search towards the best position in the search space. Two variants of the PSO algorithm have been developed, namely PSO with a local neighborhood, and PSO with a global neighborhood. According to the global neighborhood, each particle moves towards its best previous position and towards the best particle in the whole swarm, called the gbest model in the literature. On the other hand, based on the local variant so called the pbest model, each particle moves towards its best previous position and towards the best particle in its restricted neighborhood. Generally, PSO is characterized as a simple heuristic of well balanced mechanism with flexibility to enhance and adapt to both global and local exploration abilities. Compared with GA, PSO has some attractive characteristics. It has memory that enables to retain knowledge of good solutions by all particles whereas previous knowledge of the problem is destroyed once the population changes in GAs. Due to the simple concept and easy implementation, PSO has gained much attention and been successfully applied to a wide range of applications such as system identification, neural network training, mass-spring system, task assignment, supplier selection and ordering problem, power and voltage control etc. (Yoshida, 2000).

The mathematic description of PSO is presented in the followings. Suppose i denotes a particle and the dimension of the searching space is n. The position of the ith particle at iteration t in n dimensional space is represented as $X_i^t = (x_{i1}^t, x_{i2}^t, \ldots\ldots x_{in}^t)$. The pbest of the ith particle at iteration t in n dimensional space is represented as $P_i^t = (p_{i1}^t, p_{i2}^t, \ldots.p_{in}^t)$. The index of gbest i.e the best pbest among all the particles is represented by the symbol $G = (g_1^t, g_2^t, \ldots.g_n^t)$. The velocity for the ith particle at iteration t in n dimensional space is represented as $V_i^t = (v_{i1}^t + v_{i2}^t + \ldots\ldots v_{in}^t)$. PSO is initialized with a population of random solutions of the objective function. After finding the personal best and global best values, velocities and positions of each particle are updated using Eq. 11 and 12 respectively.

$$v_{ij}^{t} = w^{t-1} v_{ij}^{t-1} + c_1 r_1 (p_{ij}^{t-1} - x_{ij}^{t-1}) + c_2 r_2 (g_j^{t-1} - x_{ij}^{t-1}) \ldots\ldots\ldots(11)$$

$$x_{ij}^{t} = x_{ij}^{t-1} + v_{ij}^{t} \ldots\ldots\ldots(12)$$

where v_{ij}^t represents velocity of particle i at iteration t with respect to jth dimension (j =1,2,……n). p_{ij}^t represents the position value of the ith personal best with respect to the jth dimension. x_{ij}^t is the position value of the ith particle with respect to jth dimension. c_1 and c_2 are positive acceleration parameters, called cognitive and social parameter, respectively and r_1 and r_2 are uniform random numbers between (0,1).
w is known as inertia weight which is updated as:

$$w^t = w^{t-1} \times \alpha \ldots\ldots\ldots(13)$$

where α is a decrement factor. The parameter 'w' controls the impact of the previous velocities on the current velocity. Termination criterion might be a maximum number of iteration or maximum CPU time to terminate the search.

3.1.1 Solution Representation

One of the most important issue when designing the PSO algorithm lies on its solution representation. In order to construct a direct relationship between the problem domain and the PSO particles for the FMS loading problem, we present n number of dimensions for n number of jobs. In other words, each dimension represents a typical job. In addition, the particle $X_i^t = (x_{i1}^t, x_{i2}^t, \ldots\ldots x_{in}^t)$ corresponds to the continuous position values for n number of jobs in the loading problem. The particle itself does not present a permutation. Instead, we use the Smallest Position Value (SPV) rule to determine the sequence implied by the position values x_{ij}^t of particle X_i^t. Table 3 illustrates the solution representation of particle X_i^t for the FMS loading problem together with its corresponding velocity and sequence. According to the SPV rule, the smallest position value is $x_{i1}^t = 0.11$, so the dimension j=1 is assigned to the first job $x_{i1}^t = 4$ in the sequence; the second smallest position value is $x_{i2}^t = 0.57$, so the dimension j=2 is assigned to be the second job $x_{i2}^t = 6$ in the sequence, and so on. In other words, dimensions are sorted according to the SPV rule, i.e., according to the position values x_{ij}^t to construct the initial sequence.

Table 3 Solution Representation of Particle X_i^t in PSO

Dimension,j	1	2	3	4	5	6	7
x_i^t	1.67	2.82	1.23	0.11	3.47	0.57	0.98
v_{ij}^t	2.98	-0.87	1.51	-3.54	0.45	2.32	-1.50
Job sequence	4	6	7	3	1	2	5

3.1.2 Lack of Diversity and Mutation Operator

Particle swarm optimization schemes described above typically converge relatively rapidly in the first part of the search and then slow down or stop. This behavior has been attributed to the loss of diversity in the population and a number of researchers have suggested methods to overcome this drawback with varying degrees of success (Riget, 2002). Looking at the positions of the particles when the swarm had stagnated, it was clear that the points were very tightly clustered and the velocities were almost zero. The points were often not that far from the global optimum but the updating equations, due to the almost zero velocity, were unable to generate new solutions which might lead the swarm out of this state. This behavior can also lead to the whole swarm being trapped in a local optimum from which it becomes impossible to escape.

As mutation is capable of introducing diversity in the search procedure, two types mutation have attracted the researchers - mutation of global best and mutation based on sharing information from neighbors. Because the global best individual attracts all members of the swarm, it is possible to lead the swarm away from a current location by mutating a single individual if the mutated individual becomes the new global best. This mechanism potentially provides a means both escaping local optima and speeding up the search. Looking at the individual components of solution vectors corresponding to the global best function values revealed that it was often only a few components which had not converged to their global optimum values. This suggested the possibility of mutating a single component only of a solution vector. The latter approach introduces diversity by mutating few individuals in the swarm.

In this work, a mutation operator is introduced which mutates a position vectors of few particles selected randomly. The mutation operation is not executed in every iteration. A function, randi (0, MAXT), is used to return an integer greater than or equal to 0 and less than MAXT. In each iteration, if the elapsed time with no further progress is greater than this random value, mutation operation is executed. Two reasons may account for this strategy: 1) if there is no premature convergence happening, the execution of PSO will not be affected. 2) the algorithm will not increase computational overhead much. The mutation operator swaps between two positions of a particle randomly. Then, permutations are generated for new positions. PSO algorithm with mutation operation is as follows:

```
// t: time //
  // P: populations //
        // DELTA: the elapsed time of no further progress //
        // MAXT: maximum time of no further progress //
        t=0
        initialize P(t)
        evaluate P(t)
        while (not-termination-condition) do
        t=t+1
        update swarm according to formulae (11) and (12)
        if (DELTA > randi (0, MAXT)
        do mutation
        end if
        evaluate the swarm
        End
```

3.1.3 Proposed Algorithm

Step 1. Input total number of available machines, jobs, batch size, tool slots on each machine operations of all the job (both essential and optional), and processing time of every operation of each job.

Step 2. Initialize the parameters such as population size, maximum iteration, decrement factor, inertia weight, social and cognitive parameters. Generate initial population randomly. Construct the initial position values of the particle uniformly: $x_{ij}^{t} = x_{min} + (x_{max} - x_{min})\times U(0,1)$ where $x_{min}=0.0, x_{max}=4.0$ and U(0,1) is a uniform random number between 0 and 1. Generate initial velocities of the particle $v_{ij}^{t} = v_{min} + (v_{max} - v_{min})\times U(0,1)$ where $v_{min}=-4.0, v_{max}=4.0$ and U(0,1) is a uniform random number between 0 and 1.

Step 3. Get the initial sequence by using SPV rule. Then select the first job from that sequence and do the following:

a) First load the essential operation on the machine if and only if available machining time is greater than the time required by the essential operation otherwise reject the job.
b) Similarly load the optional operation if and only if available machining time and tool slot is greater than the time and tool slot required by the optional operation on the basis of machine having maximum available time otherwise reject the job

Step 4. Evaluate each particles fitness (System unbalance) by using equation (1) for case 1 and (6) for case 2 while satisfying their respective constraints.

Step 5. Find out the personal best (pbest) and global best (gbest)

Step 6. If no progress in pbest value is observed for an elapsed period of DELTA, carry out mutation of a particle using the mutation strategy as outlined in Section 4.2 provided DELTA is greater than a random number between zero and maximum time of no progress (MAXT)

Step 7. Update velocity, position and inertia weight by using equation (11), (12) and (13).

Step 8. Compute particles fitness similar to step 3 and find new pbest and gbest.

Step 9.Terminate if maximum number of iterations is reached and store the gbest value. Otherwise, go to Step 2.

4. Results and discussion

In real-life situations, industrial application of the model will invariably face a large number of variables and constraint. The swarm optimization based approach is proposed for taking decisions in such scenarios. The proposed PSO algorithm for the FMS loading problem is coded in Visual C^{++} and implemented in a Pentium IV PC. The performance of the PSO algorithms is evaluated by using ten benchmark problems available in open literature representing three different FMS scenarios. For solving the problems, parameters are set as population size = 25, w=0.85, α=0.9 and c1=c2=2 after a thorough examination of the results. Since different methods have been used by researchers for calculation of system unbalance, the machine loading problem of FMS has been solved in this paper by considering two cases to examine the robustness of the PSO algorithm. Computational results for case I and case II are summarized in 4 respectively using standard PSO. The results obtained by using PSO are summarized in Table 4. It may be noted that that both system unbalance and throughput can be improved if overloading of machines are permitted. However, economic justification of overloading must be looked into before permitting overloading.

Table 4 Summery of results obtained from PSO algorithm (adopted from Tiwari et. al., 2007)

Problem Number	Number of part types	Case I		Case II	
		SU	TH	SU	TH
1	7	414	22	52	47
2	6	219	52	9	64
3	6	778	46	778	46
4	8	590	137	530	137
5	6	892	96	108	120
6	5	844	86	0	125
7	10	213	107	20	122
8	12	150	159	5	148
9	8	569	128	569	128
10	14	171	112	5	130

One of the drawbacks of PSO is its premature convergence. In order to eliminate this and to improve solution quality, mutation operator is adopted from genetic algorithm. The results of proposed MPSO (PSO with mutation) are compared with four standard sequencing rules such as LPT, SPT, LIFO and FIFO in Table 5. The results indicate that MPSO improves the solution quality and outperforms other techniques available in literature in most of the instances.

Table 5 Comparison of results obtained using PSO based approach with other methods

Problem Number	SPT (SU, TH)	LPT (SU, TH)	LIFO (SU, TH)	FIFO (SU, TH)	Tiwari et. al (2007) (SU, TH)	MPSO Approach without overloading (SU, TH)	MPSO Approach with overloading (SU, TH)
1	288, 48	133, 29	158, 43	13, 44	63, 48	414,22	52, 47
2	1004, 47	187, 53	479, 62	496, 50	276, 61	219,52	9, 64
3	819, 51	778, 46	819, 51	1099, 43	819, 51	778,46	778, 46
4	1490, 107	836, 128	1490, 107	836, 128	536, 137	566,137	482, 137
5	168, 120	696, 96	1268, 110	168, 120	168, 120	892,96	108, 120
6	1540, 73	236, 97	236, 97	356, 107	356, 107	844,86	0, 125
7	776, 115	38, 118	836, 108	11, 117	20, 122	206,117	10, 127
8	960, 114	88, 148	238, 145	348, 153	9, 167	136,155	0, 173
9	1504, 113	619, 128	1049, 123	619, 128	619, 128	569,128	569, 128
10	782, 124	236, 104	246, 120	20, 112	0, 146	171,112	0, 128

An important parameter in PSO is population size. To study the effect of population size on solution quality, it is increased from 10 to 100 at the increment of 10 for test problems keeping all other parameters constant. Figure 1 shows the effect of population on system unbalance for Problem No. 1. From this figure, it can be observed that as the population size increases system unbalance decreases to certain extent and further increase of population has no effect on solution quality. This may be attributed to the fact that diversity in solution space increases as the population size increases but large increase in population size causes a random or worst point of search and increases the possibility of trapping at local optimum. However, the population size must be maintained at least two times of the number of jobs in order to obtain improved solution.

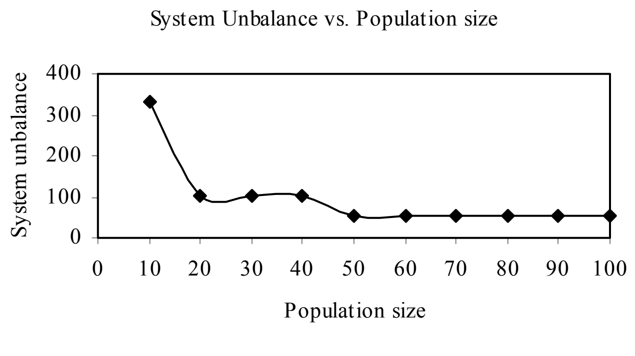

Figure 1. Best results obtained at different population size

5. Conclusions

This paper presents an efficient and reliable evolutionary based approach to solve the FMS loading problem. The proposed approach utilizes the global and local exploration capabilities of PSO to search for the optimal solution by considering availability of tool slots and machining time constraints into account. The key advantage of PSO is its computational efficiency and less parameter is required to be adjusted in order to get the optimum result compared to related techniques. Extensive computational experiments have been conducted on different benchmark problems to show the effectiveness of the proposed approach. A comparative study has been carried out for same set of problems with similar objective functions and constraints and the computational experience manifests that proposed meta-heuristic approach based on PSO outperforms the existing methodologies as far as solution quality is concerned with reasonable computational efforts. Although the objective of this study is to minimize system unbalance, the proposed meta-heuristic based on PSO not only minimizes system unbalance but also simultaneously increases the throughput for most of the instances. To avoid premature convergence, PSO algorithm is modified in this paper with the introduction of mutation operation. The performance of this algorithm is compared other related techniques. The result obtained by PSOM is promising and encouraging. It is evident from this study that overloading of machines is a viable proposition for minimizing the system unbalance but it involves cost. Therefore, a trade off between balancing of loads on machines and cost incurred must be made. In future, the study can be extended to solve loading problem by considering more realistic variables and constraints such as availability of pallets, jigs, fixtures, AGVs etc in addition to tool slots and machining time.

6. References

1. Berrada, M and Stecke, K.E. “A Branch and Bound Approach for Machine Load Balancing in Flexible Manufacturing Systems.” Management Science. 1986. p.1316–1335.
2. Kennedy, J and Eberhart, R.C. “Particle Swarm Optimization.” Proceedings of IEEE International Conference on Neural Networks. 1995. p.1942-1948.
3. Kumar, N and Shanker, K. “A Genetic Algorithm for FMS Part Type Selection and Machine Loading.” International Journal of Production Research. 2000. p.3861–3887.
4. Moreno, A.A and Ding, F.Y. Heuristics for the FMS Loading and Part Type Selection Problems.” International Journal of Flexible Manufacturing System.1993. p.287–300.
5. Mukhopadhyay, S.K, Midha, S and Muralikrishna, V. “A Heuristic Procedure for Loading Problems in Flexible Manufacturing Systems.” International Journal of Production Research .1992. p.2213-2228.
6. Nagarjuna, N, Mahesh, O and Rajagopal, K. “A Heuristic based on Multi-Stage Programming Approach for Machine-Loading Problem in a Flexible Manufacturing System.” Robotics and Computer Integrated Manufacturing. 2006. p.342-352.
7. Riget, J and Vesterstroem, J.S. “A Diversity Guided Particle Swarm Optimiser - the ARPSO.” Dept. of Computer Science, University of Aarhus, Technical Report. No. 2002-02 EVA Life. 2002.
8. Shanker K, Srinivasulu A. “Some Solution Methodologies for Loading Problems in a Flexible Manufacturing System.” International Journal of Production Research. 1989. p.1019–1034.
9. Stecke, K.E. “Formulation and Solution of Non-Linear Integer Production Planning Problem for Flexible Manufacturing System.” Management Science. 1983. p.273–288.
10. Swarnkar, R and Tiwari, M.K. “Modeling Machine Loading Problem of FMSs and its Solution Methodology is using a Hybrid Tabu Search and Simulated Annealing-Based Heuristic Approach.” Robotics and Computer Integrated Manufacturing. 2004. p.199–209.
11. Tiwari, M.K, Hazarika, B, Vidyarthi, N.K, Jaggi, P and Mukhopadhyay, S.K. “A Heuristic Solution Approach to the Machine Loading Problem of an FMS and its Petri Net Model.” International Journal of Production Research.1997. p.2269-2284.
12. Tiwari, M. K., Saha, J and Mukhopadhyay, S. K. “Heuristic Solution Approaches for Combined-Job Sequencing and Machine Loading Problem in Flexible Manufacturing Systems.” International Journal of Advanced Manufacturing Technology. 2007. p.716–730.
13. Vidyarthi, N.K and Tiwari, M.K. “Machine Loading Problem of FMS: A Fuzzy-Based Heuristic Approach.” International Journal of Production Research. 2001. P. 953–979.
14. Yoshida, H, Kawata, K, Fukuyama, Y and Nakanishi, Y. “A Particle Swarm Optimization for Reactive Power and Voltage Control Considering Voltage Security Assessment.” IEEE Transactions on Power Systems. 2000. p.1232-1239.