STUDIES ON JOB SHOPS FOR OPTIMUM SCHEDULING, LOT SIZING AND INTEGRATION USING EVOLUTIONARY METHODS

Submitted in partial fulfilment of the requirements

for the award of the degree of

Doctor of Philosophy

by

SUDHIR KUMAR MISHRA

Roll No: 700913

Supervisor:

Dr. C.S.P. RAO
Professor

Department of Mechanical Engineering
NATIONAL INSTITUTE OF TECHNOLOGY
WARANGAL – 506004
Telangana State, INDIA.
December – 2016

THESIS APPROVAL FOR Ph.D.

This thesis entitled **"Studies on Job Shops for Optimum Scheduling, Lot Sizing and Integration Using Evolutionary Methods"** by **Mr. Sudhir Kumar Mishra** is approved for the degree of Doctor of Philosophy.

Examiners

_____

_____

_____

Supervisor(s)

Dr. Dasharath Ram Yadav
(External Supervisor)

Outstanding Scientist, D.R.D.L. Hyderabad

Dr. C. S. P. Rao

Professor, Mechanical Engineering Department, NIT Warangal

Chairman

Prof. S. Srinivasa Rao

Head, Mechanical Engineering Department, NIT Warangal

NATIONAL INSTITUTE OF TECHNOLOGY

WARANGAL – 506 004, Telangana State, INDIA

CERTIFICATE

This is to certify the thesis entitled **"Studies on Job Shops for Optimum Scheduling, Lot Sizing and Integration Using Evolutionary Methods"** submitted by **Mr. Sudhir Kumar Mishra**, Roll No. 700913, to **National Institute of Technology, Warangal** for partial fulfilment of the requirements for the award of the degree of **Doctor of Philosophy in Mechanical Engineering** is a record of bonafide research work carried out by him under our supervision and guidance. This work has not been submitted elsewhere for the award of any degree.

Dr. Dasharath Ram Yadav
(External Supervisor)
Outstanding Scientist,
D.R.D.L. Hyderabad

Dr. C.S.P. Rao
(Supervisor)
Professor,
Department of Mechanical Engineering,
National Institute of Technology,
Warangal, Telangana State.

Place: Warangal.

Date:

NATIONAL INSTITUTE OF TECHNOLOGY
WARANGAL – 506 004, Telangana State, INDIA

DECLARATION

This is to certify that the work presented in the thesis entitled "**Studies on Job Shops for Optimum Scheduling, Lot Sizing and Integration Using Evolutionary Methods**", is a bonafide work done by me under the supervision of **Dr. C.S.P. Rao**, Professor, Department of Mechanical Engineering, NIT Warangal and **Dr. Dasharath Ram Yadav**, Outstanding Scientist, D.R.D.L. Hyderabad, India and has not been submitted for the award of any degree to any other University or Institute.

I declare that this written submission represents my ideas in my own words and where ever others ideas or words are included have been adequately cited and referenced with the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will cause for disciplinary action by the institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Place: Warangal.
Date:

Sudhir Kumar Mishra
Roll No. 700913

Dedicated to

Missile Man of India,

honourable Bharat Ratna,

Late Dr. A.P.J. Abdul Kalaam

ACKNOWLEDGEMENTS

It is beyond my capacity to express adequately my gratitude and respect to my beloved and respected guru ***Prof. C.S.P Rao***, Professor, Department of Mechanical Engineering, National Institute of Technology, Warangal, for his deep involvement and invaluable guidance throughout the course of this research work. His encouragement, meticulous attention and his insistence on clarity and thoroughness are evident at every stage of this research work. I shall ever remember him as a source of inspiration and gives me unbounded pleasure to express my sincere thanks to him and as a source of direction and encouragement to me in pursuing my long-cherished ambition of attaining higher academic levels.

I sincerely acknowledge the friendly and motivating words of my co-guide ***Dr. Dasharath Ram Yadav***. His frequent telephone calls to know the progress of research work used to put me on edge to strive and conclude the research in time.

I like to extend my sincere thanks to **Prof. S. Srinivasa Rao**, Head of the Mechanical Engineering Department, **Dr. A. Neelakanteswara Rao**, Section Head, Production Engineering Section and all the members of my Doctor Scrutiny Committee **Prof. L. Krishnanand**, **Prof. N. Selvaraj**, **Dr. P.S.C. Bose**, **Prof. Debasis Dutta** - Mathematics Department and **Dr. Ritanjali Majhi**, Assistant Professor, School of Management, for all the support bestowed on me by suggesting and verifying the research work.

I express my sincere thanks to **Mr. Kishore Kumar Kandi** and **Mr. P Madhukar**, Research Scholars, Department of Mechanical Engineering, NIT Warangal and **Mrs. Hymavathi Madivada**, Ph.D, NIT Warangal for their support and help in completion of this thesis.

I express my sincere thanks to **Dr. G.R.C. Reddy**, Director (In-Charge) and **Prof. T. Srinivasa Rao**, Former Director, National Institute of Technology, Warangal for providing opportunity for my research and for the encouragement by providing necessary financial assistance for attending the conferences.

At a late age of 55 years and being a Chief Executive Officer of a hugely successful company and professional demands from a Missile Scientist, the research work was very

tasking and struggling. I wish, I could have done my doctoral work in continuation of my graduation to derive intellectual happiness. Nevertheless, so many teachers, friends and colleagues helped me in achieving one of the major milestone of my life, I sincerely thank all of them. I affectionately thank my wife ***Geeta*** for her encouragement and understanding. I remember ***Surabhi*** and ***Siddharth*** for being there with me. I would love to mention my father and mother who would be very happy to know about the completion of my research work. Their unconditional love makes me humble and confident.

I thank ***DRDO*** for permitting and sponsoring the doctoral work. Finally, I acknowledge friends, relatives, well-wishers, colleagues who have shaped me. My sincere thanks and acknowledgements.

Above all, I thank the Lord, my family and well-wishers who have been an immense support throughout my research work.

Sudhir Kumar Mishra

PUBLICATIONS

1. "A hybrid binary particle swarm optimization for large capacitated multi item multi level lot sizing problems", *Materials Science and Engineering* 149 (2016) 012040.

2. "Performance comparison of some evolutionary algorithms on job shop scheduling problems", *Materials Science and Engineering* 149 (2016) 012041.

3. "An Invasive Weed Optimization approach for Job Shop Scheduling Problems", *The International Journal of Advanced Manufacturing Technology*. (Accepted)

4. "Development of hybrid particle swarm optimization algorithm for JSSP", 3rd International and 24th AIMTDR 2010 conference at Visakhapatnam, 3-10 Dec 2010.

5. "Application of memetric algorithm for job shop scheduling problems", National conference on recent advances in manufacturing technology at Gunupur, Orissa, 10 Jan 2010.

6. "A hybrid PSO approach for job shop scheduling problems", National conference on recent advances in manufacturing technology at Gunupur, Orissa, 10 Jan 2010.

ABSTRACT

Over the past few years a lot of research has been done on Evolutionary Algorithms which mimic biological process to find an optimum solution for combinatorial problems. They employ a probabilistic search for locating a globally optimal solution. They have many advantages. They are robust in the sense that they provide a set of solutions near the optimal one on a wide range of problems. They can be easily modified with respect to the objective functions and constraints. Evolutionary algorithms such as Tabu Search, Simulated Annealing, Particle swarm optimization, Generic algorithms, Ant colony optimization have been successfully applied to JSSP and good results were obtained. Several works related to JSSP published in journals on both traditional and non-traditional algorithms have been thoroughly reviewed and there is always a scope for developing new methods and algorithms for improvement of computational efficiency and quality of solution.

In this work, the author has used particle swarm optimization and binary particle swarm algorithms. The hybrids of these algorithms namely HPSO and HBPSO were developed. All the algorithms were applied on:

i. Job shop scheduling problems
ii. Lot sizing problems
iii. Integrated lot sizing and scheduling problems.

The codes were developed in MATLAB and run on intel core 2 DeoT6400@2.0 GHz computer. Each problem was made to run 30 times on each algorithm to reduce redundancy.

250 bench mark problems of JSSP available in the literature were thoroughly tested using PSO, HPSO, BPSO and HBPSO algorithms. The problems were tested with different sizes of initial population. After thorough testing, we found that the population size of initial pod is best equal to number of operations of n x m problem for fast convergence to an optimal solution. In other situations, the chances of getting local optimal solutions are more. Simulated annealing and iterative improvement methods were combined in the execution of PSO and BPSO algorithms and thus HPSO and HBPSO algorithms were developed.

Ten tough problems given by Yamada, 1997 i.e. ABZ7, ABZ8, ABZ9, LA21, LA24, LA25, LA29, LA38 and LA40 were also solved by PSO, HPSO, BPSO and HBPSO and it

was observed that BPSO, HPSO and HBPSO algorithms produce BKS values for these tough problems. Hence, the algorithms developed are robust and able to solve any kind of JSSPs effectively. PSO was hybridized with simulated annealing and the resultant algorithm is called Hybridized PSO (HPSO). HPSO shows an improvement of 42% compared to PSO. Similarly, HBPSO is produced by applying iterative improvement in BPSO. HBPSO produces 12% improvement over BPSO. We also found that HBPSO and HPSO are fast converging as compared to BPSO and PSO. Hence, they emerge as time and solution efficient algorithms. These algorithms are recommended to solve any kind of Job shop scheduling problems.

An attempt is successfully made to implement Binary Particle Swarm Optimization (BPSO) algorithm and its variant HBPSO to address the integration of lot sizing and scheduling problems that arise in a typical manufacturing industry. Scheduling decisions are respected during the decision taken for lot sizing problems by taking a capacity constraint similar to the known scheduling problem. We have integrated the problem up to some extent which leads to the reduction of cost involved in Lot sizing problems. The proposed algorithm can be applied to any type of production system and product structure. BPSO/HBPSO technique has been successfully implemented for integration of different LS problems like single-item, multi-level (SIML) and single-item, single-level (SISL), multi-item, multi-level (MIML) problems with three product structures with a scheduling constraint. We found reduction in inventory cost by considering the scheduling constraint. We have tested a very few problem instances. However, we have to test thoroughly a large number of scheduling constraints in lot sizing problem for effective evaluation of integration of scheduling and lot sizing. We found that convergence of solution takes place at large number of iterations and sizes of swarm. The author emphasizes that there is no work reported on the integration problem using BPSO/HBPSO technique in the contemporary literature.

Computational experience shows that HPSO and HBPSO algorithms can be implemented as separate optimization modules for solving all types of JSSP and Lot Sizing Problems independently in SAP/MRP-II packages.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

ABZ	Adams Balas & Zawak
ACO	Ant colony optimization
AI	Artificial Intelligence
AIA	Artificial Immune Algorithm
AIA	Artificial Immune Algorithm
ANN	Artificial Neural Networks
B-B	Branch and Bound
BCO	Bee Colony Optimization
BEP	Back-Error propagation
BFO	Bacterial Foraging Optimization
BKS	Best Known Solution
BW	Band Width
CAR	Jacques Carlier
COP	Constraint Optimization Problem
CPU	Central Processing Unit
CS	Cockoo Search
DE	Differential Evolution
DMU	Demirkol, Mehta & Uzsoy
DP	Dynamic Programming
EA	Evolutionary Algorithms
EC	Evolutionary Computation
EDD	Earliest Due Date First
EMO	Evolutionary Multi-Objective Optimization
FA	Firefly Algorithm
FCFS	First Come First Served
FISFS	First in System First Served
FL	Fuzzy logic
FT	Fisher and Thompson
GA	Genetic Algorithm
GLS	Genetic Local Search

GRASP	Greedy Random Adaptive Search Procedure
HAIA	Hybrid Artificial Immune Algorithm
HIA	Hybrid Intelligence Systems
HMCR	Harmony Memory Considering Rate
HPSO	Hybrid Particle Swarm Optimization
HS	Harmony Search
I/O	Input/Output
IMBHS	Improved Music Based Harmony Search
IWO	Invasive Weed Optimization
IWO	Invasive Weed Optimization
JSSP	Job Shop Scheduling Problem
LA	Lawrence
LPT	Longest Processing Time
LSF	Least Slack First
LWR	Least Work Remaining
MA	Mimetic algorithm
MBHS	Music Based Harmonic Search
MINDD	Minimum Due Date
MOPNR	Most Operations Remaining
MWR	Most Work Remaining
NI	Number of Improvisations
NP	Non Polynomial
ORB	Applegate and Cook
P	Polynomial
PAR	Pitch Adjusting Rate
PDR	Priority Dispatch Rules
PSO	Particle Swarm Optimization
PSO	Particle Swarm Optimization
RAND	Random
RP	Random Population
SA	Simulated Annealing
SBP	Shifting Bottleneck Procedure
SBP	Shifting Bottleneck Procedure

SD	Surrogate duality
SP	Selective Population
SPT	Shortest Processing Time
SWV	Storer, Vaccari & Wu
TA	Taillard
YN	Yamada and Nakano

NOMENCLATURE

n	Number of jobs
m	Number of machines
j	Subscript to a job
i	Subscript to a machine
o	Operations to be scheduled
Gj	Operating sequence
	Processing time
	Due date
Prec	Precedence constraints
brkdwn	Breakdowns
Cmax	Makespan
L j	Lateness
	Tardiness
Lmax	Maximum Lateness
Cj	Completion Time
Fj	Flow Time
WIP	Work-in-process inventory
Pbest	Particle best
Gbest	global best
	Inertia weight
	Maximum inertia weight
	Minimum inertia weight
	Acceleration constants
xij	Particle position
*Xij**	Updated particle position
Sp	All the particles position vector
*Sp**	Updated all the particles position vector
So	Operation sequence
*So**	Updated operation sequence
δ	Change in objective function
T	Current temperature

r	A random number uniformly distributed between 0 and 1.
Ab	Antibody
	Concentration of the antibody
W	Potential solution weed
N	Number of seed
Max	Max no of seeds a weed can have
w	Worst rank
b	best rank
i	Fitness of the weed considered
Sin	Fnitial standard deviation
Sfin	Final standard deviation
S	Standard deviation
n	Non-linear index
p	Dimension of the search space
S	Total number of bacteria in the population
Nc	The number of chemotactic steps
Ns	The swimming length
Nre	The number of reproduction steps
Ned	The number of elimination-dispersal events
Ped	Elimination-dispersal probability
C (i)	The size of the step taken in the random direction specified by the tumble vector in the random direction

CHAPTER 1

INTRODUCTION TO SCHEDULING

1.1. Introduction

There is a conflicting requirement between the demand and the supply sides of present day supply chain. A typical present day customer demands a wide variety of product at a low cost, whereas a company fulfilling this demand faces an ever-increasing competition which translates finally to reducing the cost of production. As an example, zero inventory - rapid response is one set of conflicting requirement faced by the supply side of current supply chain. These conflicting requirements require accurate, effective and efficient scheduling which is practically a complicated task in any of the production environments. Hence, this necessitates the need for an effective heuristics and scheduling algorithm.

The primarily goal of Scheduling is to resolve a Constraint Optimization Problem (COP). Any manufacturing process requires an optimized allocation of resources (which are competing for allocation). The process of scheduling achieves this by identifying a sequential resource allocation so that a specific objective function is optimized. Scheduling is planning of time based activities that involves allocation of scarce resources by optimizing one or more performance measures. Activities and resources can take on various forms depending on the situation.

Resources in an assembly may be machines. For, example resources in a computer hardware plant are CPU, memory, and I/O devices. Similarly, mechanics are one of the resources in an automobile repair shop and runways are resources at an airport. These processes may involve numerous operational activities, for example landings and take-offs at an airport, execution of a computer programme or repairing a vehicle in an automobile repair shop, and so on. Optimization can be measured through different parameters such as by minimizing the makespan, or by minimizing mean flow time and mean tardiness. A manufacturing process with a good scheduling algorithm can reduce production cost and empower the company to stay competitive in the market.

In the 1950s, scheduling researchers in the areas of industrial engineering, operations research, and management juggled with the problem of managing different activities happening at a workshop. In the late 1960s, Computer Scientists too encountered problems pertaining to scheduling while developing operating systems. In those days, computational resources were scarce and effective utilization of those scarce resources could have reduced the cost of computer programmes execution and this therefore, provided an economical reason to study scheduling.

Problems pertaining to scheduling were relatively simple when studied in the years around 1950s. To provide optimal solution, a number of efficient algorithms were developed. Some of the well-known ones include Jackson's Structured Programming, Johnson's algorithm, and Smith-Waterman algorithm. Scheduling problems in the later period became more complex/ sophisticated and the researchers were unable to develop any effective algorithm to solve such problems. Most of the researchers studied to develop efficient branch and bound methods that were, in essence, exponential time algorithms. As the complexity theory advanced, the researchers realized that a lot of these problems could be quite difficult when attempted for solution. Many scheduling problems, in the 1970s, were found to be NP-hard.

Different research directions were pursued in the industry and academia in the 1980s. For example, in approximation algorithms, one of the direction was development and analysis while the other direction was increased attention towards stochastic scheduling problems. After that, research in scheduling theory went through a rapid progress.

1.2. Scheduling Terminology and Definitions

In all problems related to scheduling, finite number of machines and jobs are taken, which are represented by "m" and "n" respectively. The subscript "i" and "j" usually represent machine and job respectively. In case, where job undergoes multiple operations, the terminology (i, j) represents job j on the machine i.

The below mentioned terminology is associated with scheduling of n jobs on m machines.

Processing time: This is the duration of a job (represented by j) to get processed on a machine (represented by i). "i" is not mentioned in case when the processing time of the job is independent of machine or, if it is not to be processed on multiple machines.

Due date: Job j due date is referred as "dj". It is the promised date of completion. The job incurs a penalty for the same, if it is not completed within due date. When meeting "the due date" is a must, the due date is referred to as a deadline.

Precedence constraints (Prec): This constraint may exist in one/ more than one machine environment, when there is a requirement that prior to the start of processing of another job, one or more jobs may have to be completed. There are many special cases of precedence constraints. The constraints are mentioned as chains when the maximum number of successors and predecessors for every job is one.

Breakdown (brkdwn): Machine breakdown implies that machines are not continuously available.

Makespan (Cmax): It is defined as max (C1,C2….Cn). It is the duration between the time of completion of last job getting over and the start of the first job. High utilization of machine(s) is usually implied from a minimum makespan of the machine(s).

Makespan can be calculated as, Makespan = Time of completion of last job - Starting time of first job.

Lateness (Lj): Lateness is a measure of the difference between the completion time of a task and its due date. A task having "Positive lateness" means the task was completed after its due date, whereas if the task has negative lateness, the task was completed before its due date.

Tardiness: This is the measure of positive lateness. When a task is completed early, the task will be having negative lateness but zero tardiness. When a task has positive lateness, it has equal positive tardiness also. Tardiness is measured as the maximum of {0, L j}.

Maximum Lateness (Lmax): It is defined as max (L1, L2…, Ln). It measures the extreme cases of due date violation.

Slack: Slack can be measured as the time difference between the remaining time of a task's due date and its processing time.

Completion Time (Cj): This is the span between the time at which work begins on the first job, which is referred to as t = 0, and the time when a task j is completed (finished). This span is denoted as Cj.

Flow Time (Fj): This is the duration between the point of time at which a task is available for processing and the point (of time) at which it is finished.

Flow time = Processing time + the time the task waits before being processed

Work-in-process inventory: It is a job which has not yet undergone all the operations supposed to be performed on it. Measure of WIP can be expressed as individual units, number or quantity of jobs, monetary value for entire system, in terms of weeks of supply etc.

Total inventory: This represents the sum of total receipts and the inventories in hand. This could be expressed in weeks of supply, dollars, or units (individual items only).

Utilization: Utilization = The number of hours the worker has worked or the machine is utilized in giving certain productive output. It can be measured as a ratio between Productive work time and total work time available.

Heuristic: Heuristics are the problem-solving procedures or rules of thumb which have been shown to produce good results. Heuristics, however, cannot guarantee optimal results.

1.3. Classification of Scheduling Problems

Scheduling problems have been traditionally studied as cases involving single and parallel machines, flow and Job shop problems.

1.3.1. Single Machine Scheduling

It aims to find out the best schedule for single job or jobs in batches. In single machine scheduling, only one resource is utilized to process the jobs. Typically, 'best', implies a schedule that is able to minimize some performance metric or metrices of a selected number of performances. For example, minimizing one or more or all of the parameter(s) such as (i) total tardiness, (ii) delay in job completion or (iii) Total time taken to finish all the operations to be performed on the job. In this case, an "early/tardy problem" is the problem for which the objective is to identify a schedule which is able to minimize the cost due to total earliness/ tardiness of all the jobs.

A single machine environment, which is one of the least complicated, is considered as a special case for rest of the machine environments. The result obtained for single machine models provides a basis for heuristics approach even for cases with complex machine environments.

Scheduling problems which are encountered in a complex machine environment are often broken down into sub problems to reduce the problem to the levels to make it equivalent to working on a single machine.

1.3.2. Parallel Machine Scheduling

This is one of the most common problems encountered in real world, where jobs are required to be processed with an objective to minimize makespan, total processing time, or operational cost. In practical scenario, a scheduler often deals with the situation where loads are to be balanced (machines in parallel). A good schedule can be ensured by minimizing the makespan.

1.3.3. Flow Shop Scheduling

In describing a flow shop scheduling, the shop has m different machines, and every Job be processed on 'm' machines in natural machine order.

The numbering of the machines can be done in the order of the job sequence with lower machine number indicating the preceding operation performed by the machine than the higher machine number (which performs the next operation). Figures 1.1 and 1.2 shown below depict an example of flow shop:

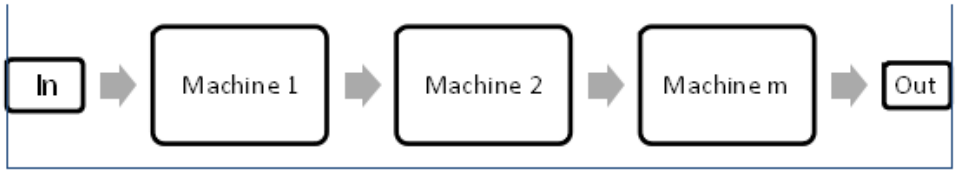


Figure 1.1: Pure Flow Shop

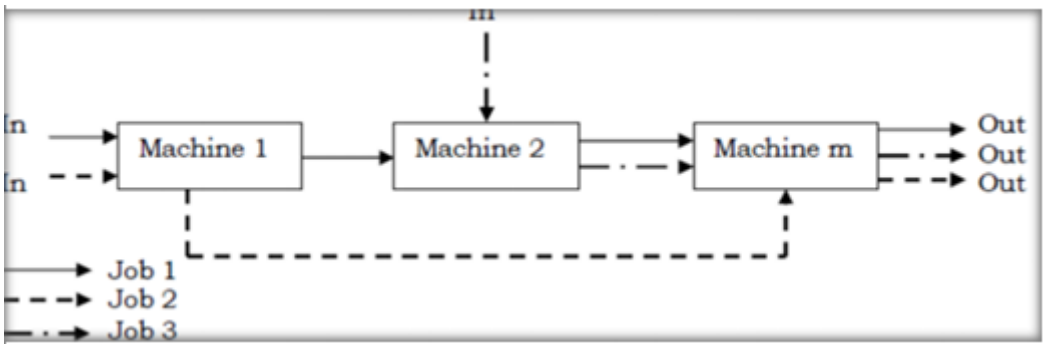


Figure 1.2: General Flow Shop

Figure 1.1 depicts a "pure" flow shop in which all the jobs are processed on all machines in a sequential order with one operation on one machine each. Figure 1.2 depicts a General Flow Shop in which a number of jobs are processed through different machines. It also shows that their operations may not always require adjacent machines in the numbered sequence.

1.3.4. Job Shop Scheduling

JSSP is scheduling of Jobs on finite set of machines in a preferential order of operations of each Job on finite machines so as to optimize the performance of machine (s). Each operation requires one or several machines in order to be successfully performed, and there can be several machines of the same type so that alternate machines can be utilized for each of the operation. Operations are atomic or uninterruptable i.e. once initiated, they cannot be stopped. In the simplest case, each Job is having a fixed route, and each resource is able to process only one operation at a time. In job shop case, it is more apt to describe an operation with the triplet (i, j, k) so as to represent the full description of a job i.e. operation denoted by "I" of the job "j" requires machine "k". A measure of performance must be specified to complete the problem statement.

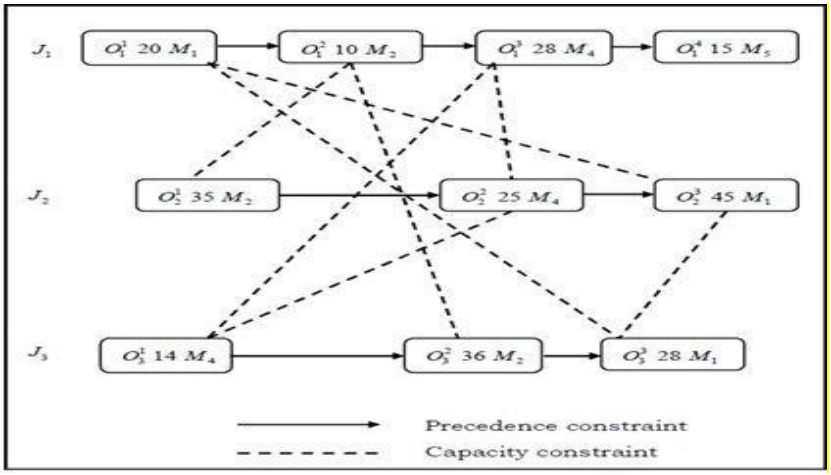


Figure 1.3: A typical job shop with 3 jobs

The problem is then to construct a feasible schedule and then optimize the performance measure. The constraints are specified by routing of the sequence of operations to be performed on the job through various machines. There is resource capacity constraint also which prohibit resources to perform two or more operations at one time. Figure 1.3 shows an example of a job shop problem. Each operation is labeled by a triplet containing operation description, operation duration

(example 20) and resource requirement. The arrows in the figure represent precedence constraints.

A job is processed by its route with time and priority. The amalgamation of Jobs will be a key issue while scheduling jobs in a job shop. In the Figure 1.3, {J1, J2, J3} represent some of the parts to be manufactured in the Job Shop. Job Shop Scheduling Problem (JSSP) deals with sequencing all these operations in such a way that:

i. No one job is pre-empted.
ii. The precedence given by the technical sequence is respected.
iii. No two jobs are processed in parallel on a single machine.
iv. Each job is completely processed, although the jobs may wait or get delayed during the operations.
v. No release time exists i.e. the Jobs may start at any time.
vi. Jobs must wait for the next machine to be available.
vii. More than one operation at a time on a single machine is not allowed.
viii. The operations set-up times are not dependent on sequence; the processing time includes set up time.
ix. Each type of machine is unique i.e. only one of each machine type exists.
x. Idling of machines may exist within the schedulable period.
xi. At any point of time, machines are available.

A Job shop deals with both low/ medium volume production. Job shop scheduling deals with the sequencing of processing jobs, where each job is to be processed in different sequence, on different machines or stages. Each operation (except the first and last) has one or more predecessor and successor.

In a scenario of JSSP having n jobs. These n jobs must be processed on m machines and each job is characterized by a chain of operation (in a typical given sequence) on set of machines. Each job is to be processed in an uninterrupted time period of a specified length. At the most, at a particular time, one machine is capable of handling only 1 job.

Essentially, the solution to this problem is to find out when to schedule each operation of a job on each machine so as to minimize the last operation's finishing time which is also termed as makespan. To explain this case in a more specific way, set of jobs [J_1, J_2, J_2,…., J_n], set of

machines [M_1, M_2, M_3,…., M_m] and set of operations to be scheduled [O_1, O_2, O_3, …, $O_{n\times m}$, $O_{n\times m+1}$] are defined. The operations are interrelated by two constraints–precedence constraint & capacity constraint. In the precedence constraint, once all predecessor operations are finished, then and only then each operation "j" must be scheduled. In capacity constraint, for scheduling an operation "j", the required machine must be idle. As already explained, the duration in which all operations for all jobs are completed is referred to as the makespan. As specified earlier, in JSSP, the objectives which are to be considered are minimizing makespan, mean flow time, and mean tardiness while satisfying all the capacity and precedence constraints present in the system.

The total number of all possible schedules (both feasible and infeasible) is (n!)m for the problems where there are n jobs to be processed on m machines. Here, it can be seen that it is practically very difficult to go through all the alternatives in order to find the best solution even for small nxm problem. For example, Fisher-Thompson benchmark problems are 10 × 10 and 20 × 10. The search space sizes for these benchmark problems are about 3.96 × 1065 and 7.2651 × 10183 of possible solutions respectively.

The practical difficulties of evaluating all options for identifying feasible schedules, with the optimal one, have been explained in the preceding texts. Most of the JSSP cases are also NP-hard/ NP-hard (strong) thus making the problem as very hard member of this class (Nakano and Yamada, 1991; Lawler et al., 1993). There is no strategy devised so far that can guarantee optimal solutions for JSSP instances larger than 20 × 10.

1.4. Scheduling Methods

Researchers have used various methods in the field of Job Shop Scheduling. They are broadly categorized in 2 groups: (i) methods using "traditional approaches" and (ii) methods using "non-traditional approaches".

1.4.1. Non-Traditional Techniques

These techniques have been found to be quite agile. However, these techniques generally don't produce an optimal solution. These methods include:

i. Constructive Methods, which uses a set of heuristic, priority and composite dispatch rules.
ii. Insertion Algorithms such as heuristics using bottleneck and Shifting Bottleneck rules etc.
iii. Evolutionary algorithms such as GA, PSO etc.
iv. Local Search Techniques includes Hill climbing, Simulated Annealing, and Tabu Search, problem etc.
v. Iterative Methods such as Artificial Neural Network etc.
vi. Heuristics Procedure such as Invasive Weed Optimization, Harmony Search, Bacterial Foraging Optimization, Intelligent Water Drops, Firefly Algorithms and Hybrid Techniques.

A number of practitioners have given a lot of attention to heuristics approaches as these can often provide high-quality solutions taking reasonable computation time instead of finding optimal solution which takes a lot of computation time.

1.4.1. 1. Heuristic Dispatch Rules

i. Earliest Due Date First (EDD): In this rule, priority is given to the job which is having the earliest due date.

ii. First in System First Served (FISFS): In this rule, priority is given to the job that arrived in the shop first (not on the machine).

iii. **First Come First Served (FCFS):** Priority be given to those jobs that arrived at the machine first.

iv. Least Slack First (LSF): In this rule, priority is given to the processing of the job (among several jobs) which is having the least slack. Slack has already been explained earlier.

v. Shortest Processing Time (SPT): In this rule, priority is given to the job which is having the shortest processing time on a machine and thus Jobs are allotted based on SPT. As this method speeds up the progress of several short jobs at the expense of a few long jobs, on the whole, the average flow time is reduced, but long jobs may take very long waiting times.

vi. Longest Processing Time (LPT): In this method, the job which requires longest processing time is prioritized on the machine under consideration.

vii. Least Work Remaining (LWR): In this method, the job which requires least amount of total remaining processing is prioritized.

viii. Most Work Remaining (MWKR): In this method, the job which requires most amount of total remaining processing time is prioritized.

ix. Most Operations Remaining (MOPNR): In this method, the job having largest number of successor operations is prioritized.

x. Shortest ratio of processing time to Total processing time: In this method, a part is selected first, which has the least ratio of processing time and the total processing time.

1.4.1.2. Insertion Algorithms

i. Shifting Bottleneck Based Heuristics

This heuristic can be regarded as one of the most successful Job shop scheduling heuristic procedure developed. M is representing the set of all m machines in the shifting bottleneck procedure. While describing an iteration of shifting bottleneck procedure, it is assumed that in previous iterations, a selection of disjunctive arcs has been determined for a subset M0 of machines i.e. a job sequence for each of the machines in M0 is already specified.

To decide the machine that should be included next in M0, determination of unscheduled machine is attempted which causes the severest disruption. In order to determine the machine that should be included in next step in the set M0, all the disjunctive arcs of these machines are deleted and the original directed graph is hence modified. This deletion activity implies that all associated operations which were previously expected to be done in series (one after another) can now be performed in parallel.

The obtained graph has critical paths, which can be one/ more that decide the corresponding makespan. This is repeated in order to include an additional machine to the current set in subsequent iteration.

ii. Beam Search

Branch and bound principles are the basis of this method. In present time, Enumerative branch and bound is one of the extensively utilized methods for NP-hard scheduling problems. The major demerits of B&B are that the method becomes time consuming if the number of nodes for consideration is very large.

In Filtered beam search, which is an adaption of B&B, all the nodes are not evaluated at any given level. The most promising nodes at a level k are selected. These are treated as nodes to branch from, the other remaining nodes are eliminated (discarded) permanently at each level. The quantity (numbers) of retained nodes is known as beam width of the search. Evaluation process of this method determines the promising nodes. However, evaluation of each node and then obtaining an estimate for the possibility of its offspring consumes a considerable amount of time. To make the process fast and closer to better solution, a filter is applied i.e. a crude prediction is done for all the nodes generated at level "k". Based on the outcome of the previous step, a few number of nodes are selected for evaluation, and the left-over nodes are permanently disposed-off. The quantity (or no.) of nodes thus selected is referred to as the filter width. The results of all nodes that passed filter gives the way for selection of a nodes subset for generation of further branches.

1.4.1.3. Evolutionary Algorithms

Evolutionary algorithms are stochastic search methods that mimic the metaphor of natural biological evolution, i.e. natural selection and evaluation.

Evolution can be defined as the slow changes which occur in a population of organisms as a result of adaptation to its surrounding.

This theory was discovered by Darwin who coined the term "survival of the fittest". Darwin explained the theory giving example based on limited resources, struggle for existence and natural selection. He further showed that the winner (organisms) or better fitted organisms after the struggle pass on their special survival characteristics to their subsequent generation. Some of the principles of this theory were first tried for solving optimization problems in the 1960s. EA operate on the selected population of solutions by applying the survival of the fittest procedure and produce progressively better approximates to a solution. A new set of approximate solution is created in each generation according to the fitness of individual in the problem domain. Different classes of evolutionary algorithm include Genetic Algorithms,

Evolutionary Programming, Evolution Strategies, Classifier Systems, And Genetic Programming, Particle Swarm Optimization, Ant Colony Optimization, Bee Colony Optimization, Artificial Immune Algorithm, Invasive Weed Optimization, Bacterial Foraging Optimization, Harmony Search, Intelligent Water Drops and Firefly Algorithm. All of these share a common conceptual base of simulating the Evolution of Individual structures via processes of Selection, Mutation, and Reproduction. Although they look simple from the view point of a biologist, the algorithms are sufficiently complicated for provision of robust and powerful adaptive search mechanisms.

In the present work, a similar attempt of applying existing and new evolutionary algorithms named Particle Swarm Optimization (PSO), Hybrid Particle Swarm Optimization (HPSO), Artificial Immune Algorithm (AIA), Hybrid Artificial Immune Algorithm (HAIA), and Invasive Weed Optimization (IWO), Bacterial Foraging Optimization (BFO), Music Based Harmonic Search (MBHS), Improved Music Based Harmony Search (IMBHS) to Mono and Multi Objective JSSPs has been made successfully.

i. Artificial Intelligence (AI)

AI is an area of computer science dealing with the study of how to make computers do things which, currently, are done better by humans. AI, in its quest to design intelligent systems, has not just registered modest success in developing techniques and methods for intelligent problem solving, but has fanned out to encompass a number of technologies in its fold. Some of the technologies include, but are not limited to expert systems, are neural networks, fuzzy logic, cellular automata and probabilistic reasoning. Neural networks, fuzzy logic and probabilistic reasoning are predominantly known as soft computing techniques.

ii. Artificial Neural Networks

Artificial Neural Networks (ANN) is a simplified model of the biological nervous system. ANN is basically inspired by brain in which a highly-interconnected network of a large number of processing elements called neurons are present and exhibit parallel distributed processing. Neural networks exhibit traits such as mapping capabilities or pattern association, generalization, robustness, fault tolerance and parallel and high speed information processing.

ANN learn by examples and therefore can be trained with known problem examples to 'acquire' knowledge about it. Once trained appropriately, the network can be put effectively for solving 'unknown' instances of the problem within the same knowledge domain. Neural networks have been successfully applied to problems in the fields of pattern recognition, image processing, data compression, forecasting and optimization etc.

iii. Genetic Algorithms

Genetic Algorithms (GA) are unorthodox search and optimization algorithms that mimic some of the processes of natural evolution. GA performs directed random searches through a given set of alternatives with the aim to find the best alternative in a given criteria of goodness. These criteria are expressed in terms of an objective function which is often referred to as a fitness function. Fitness is defined as a parameter representing a merit that has to be either maximized or minimized. GA have been theoretically and empirically proven to provide robust search in complex space and have found wide applicability in scientific and engineering areas including function optimization, machine learning, scheduling and others.

iv. Hybrid Intelligence Systems

Hybrid intelligence systems deal with the synergistic integration of two or more of the technologies. The main advantage is that the technologies can be selected based on their strengths to suit a specific set of problems.

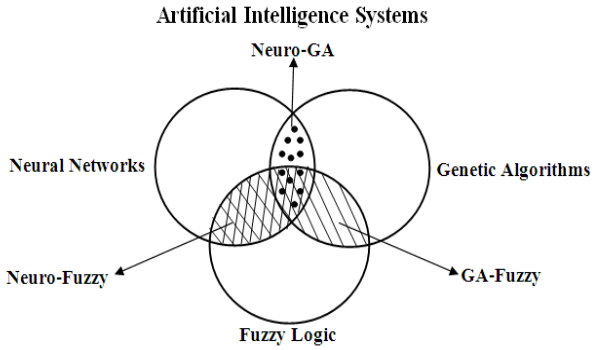


Figure 1.4 Integration of Neural Networks, Fuzzy Logic and Genetic Algorithm Technologies.

As illustrated in Figure 1.4, each of these technologies either alone or in combination can be employed for problem solving. For example, Neuro-fuzzy, GA-fuzzy, Neuro-GA, and Neuro-

fuzzy-GA technologies. Among different soft computing techniques, artificial neural networks technique is briefly discussed in the following sections.

v. Particle Swarm Optimization

Particle swarm optimization (PSO) was proposed by Kennedy and Eberhart (1995). The concept of PSO has been derived from the means of mechanism of information exchange and social behaviour of bird flocking or fish schooling. It combines local search (by self-experience) and global search (by neighbouring experience), possessing high search efficiency. PSO learned from these scenarios are utilized for getting the solutions of optimization problems. In PSO, each single solution is a "bird" in the search space called "particle". All the particles have fitness values (evaluated by the fitness function to be optimized) and velocities (which direct the flying of the particles). The particles are "flown" through the problem space by following the current optimum particles. PSO is initialized with a group of random particles (solutions) and then it searches for optima by updating generations. In each iteration, every particle is updated by following two "best" values known as pbest and gbest. pbest is the best solution (fitness) it has achieved so far (The fitness value is also stored) and gbest is the best value obtained so far by any particle in the population.

vi. Artificial Immune Algorithm

It was initially developed in 1986 by J. D. Farmer et al. in their paper called 'The Immune System, Adaptation and Machine Learning'. This was followed by another paper by G.W. Hoffman called 'A Neural Network Model Based on the Analogy with the Immune System'. AIS is a new intelligent problem-solving technique which finds applications in optimization, computer security, data clustering, pattern recognition or even fault tolerance (Dasgupta, 2006).

The primary function of the immune system is to protect the human body from attacks caused by foreign (harmful) organisms. The immune system is capable of distinguishing between the normal components of organism and the foreign material that can cause us harm called antigens (e.g., bacteria). The molecules called antibodies play the main role in the immune system response. Human body immune system generally responds to antigens through antibodies in accordance with a process called clonal selection principle (Nunes de Castro and Von Zuben, 1999). The newly-cloned cells undergo high rate of mutation or hyper-mutation

in order to increase their receptor population (called repertoire). These mutations experienced by the clones are proportional to their affinity to the antigen.

Human immune system is the basis from which models, functions and principles were derived for problem solving through Artificial Immune Systems. To comprehend the Artificial Immune Model, a knowledge of the basics of how human immune system functions, is required. It is already known that human immune system is a highly robust and adaptive system.

In human immune system, for example, a counter defensive mechanism (called antibody) attacks the infection (also known as antigen). An antigen is defined as "the sequence of jobs on a particular machine given a particular scenario" which gives the full schedule for the problem. The antigens are represented by a sequence of numbers of length for the problem. A partial schedule or an antibody is described as "a short sequence of jobs that is common to more than one schedule". The antibodies are described by the sequences of numbers of length and the length of an antigen is more than the length of an antibody. In this method, the sequence of jobs (analogous to antigens) are to be matched with partial schedules (analogous to antibodies) in an antigen universe. Before an antibody population is built, the preparation of an antigen universe is required. An antibody population is partial schedules' collection, constructed from gene libraries which consist of genotypes. All the antigens in the antigen universe constitute the gene libraries in this research. The final population (collection of best antibodies) will be the initial solutions (from the AIS model) to the local search method, when hybridized the AIS model with a local search method. Fitness is the representation of the assigned value to each antibody in the antibody population for evaluation of the coverage of an antibody over the antigens. The higher the fitness, the better is the antibody.

A detailed explanation of various steps in the algorithm is given below:

vii. Ant Colony Optimization

Ant colony optimization (ACO) metaheuristic is a novel population based approach proposed by Dorigo et al to solve several discrete optimization problems. ACO copies the methodology of how real ants are able to find the shortest route between their nest and food. It was revealed that the ants communicate through the "pheromone trail" in a swarm. A moving ant lays varying quantities of the chemical "pheromone" on the ground as it moves forward, marking

its journey by a trail of pheromone. The ants follow the path which has been traced by a large number of ants. This self-sustaining group behaviour results in the establishment of a route which is the shortest.

viii. Simulated Annealing Algorithm

Simulated Annealing (SA) is motivated by an analogy to annealing in solids. In 1982, Kirkpatrick et al (Kirkpatrick, 1983) took the idea of the Metropolis algorithm and applied it to optimization problems. The idea is to use simulated annealing to search for feasible solutions and converge at an optimal solution. The algorithm is based on the simulation of cooling of material in a heat bath (annealing process). The structural properties of a solid depend on the rate of cooling if it is heated past melting point and then cooled. When the rate of cooling of the liquid is slow enough, then large crystals are formed. However, if the rate of cooling of the liquid is quick (commonly referred to as quenching), the crystals formed will be having imperfections. Metropolis's algorithm simulates the material as a system of particles. The algorithm simulates the process of cooling by gradually lowering the system temperature until it reaches a steady, frozen state. Relationship between physical annealing and simulated annealing is shown in the following table:

Thermodynamic Simulation	Combinatorial Optimization
System States	Feasible Solutions
Energy	Cost
Change of State	Neighboring Solutions
Temperature	Control Parameter
Frozen State	Heuristic Solution

Table 1.1: Relationship between physical annealing and simulated annealing

Any combinatorial optimization problem can be converted into an annealing algorithm using these mappings. The major advantage of SA over other methods is its ability to avoid becoming trapped at local minima. The algorithm employs a random search which accepts changes that decrease objective function, "f", and also some changes that increase it.

ix. Memetic algorithm

This algorithm combines local search (heuristic) and global search (population based) made by each of the individual. In this way, the combination of two non-traditional techniques produces an evolutionary algorithm. This algorithm is basically a population based approach.

This approach of combining has been identified as a powerful algorithmic paradigm for evolutionary computing since it comprises of a separate local search process to refine individuals. When compared with EA, the relative advantages of MA have been quite consistent in complex search spaces.

x. Tabu search

The basic idea of Tabu search (Glover 1989, 1990) is to explore the search space for all feasible scheduling solutions by a sequence of moves. Just like gradient-based techniques, a move from one schedule to another is made by evaluating all candidates and choosing the best available one. Some of the moves are classified as tabu (i.e., they are forbidden) because they either trap the search at a local optimum, or they lead to cycling (repeating part of the search). These moves are listed as/ put onto something called the Tabu List, built from the history of previous moves. These Tabu moves force exploration of the search space until the old solution area (e.g., local optimum) is left behind. Here, another key element is that of freeing the search by a short-term memory function that provides "strategic forgetting".

1.5. Inventory Control

In production/ manufacturing system, the end products are usually made up of many intermediate items. The intermediate items usually consist of purchased parts and raw materials. The end product or the finished good is described by a bill of material (BOM), which is the recipe of product. The product structure describes the number of levels in the BOM and for a complex product, the number of levels are more (multi-level structures). In view of this, satisfying external and inter demands becomes more complex. The Lot Sizing Decision therefore becomes very important. Lot size might be the amount of production or purchase quantity depending on the demand at different time buckets to ensure and satisfy customer requirements. Minimizing total production cost is always a tradeoff decision between ordering and holding cost. Here, the order quantity in a particular period may be (i) requirement of that period or (ii) requirement of that period including group of requirements of periods ahead or (iii) zero.

Lot sizing problems are primarily divided into two types: (a) single level lot sizing (SLLS), and (b) multi-level lot sizing (MLLS). The number of final products (finished goods) and

capacity (capacitated/ uncapacitated) also affect the modeling and complexity of production planning problems.

1.5.1. Production Planning

Production Planning is a systematic tool that aims to fixing of production goals and accordingly estimation of resources that are required to achieve these goals. It is a plan that takes into consideration the best use of production resources in order to satisfy production goals (meeting production requirements and anticipating sales opportunities) over certain period named the "planning horizon". The aim of Production Planning is to come out with a detailed strategy to achieve the production goals in timely, economically and efficiently manner. And for that, the plan makes a forecast of each step in the production process, including the problems that could arise during the production process. The planning phase of production thus tries to remove any such problems and also attempts to remove the causes of wastage.

Some of the well-known types of production planning include job- or project-based planning, batch planning and continuous or mass production planning. The Job based planning is done to cater to the needs of a single or small customer.

1.5.1.1. Planning Characteristics

Different business strategies are devised based on the duration framework which affects the planning characteristics. These have been explained below:

i. Short-Term

In short-term planning, the existing capabilities of a company are taken into consideration, and strategies are devised accordingly to improve them. For example, the skills of the employees and their work attitudes, or the present condition of production equipment or product quality etc. can be classified as short-term concerns. Short term solutions are then put into place to address these issues. For example, imparting training to the employees, servicing the production tools/equipment, and fixing the product quality are short-term solutions. Here, it is pertinent to note that the solutions provided for short term plans set the stage to address the problems in a more comprehensive manner in the longer term.

ii. Medium-Term

Such planning provides a more permanent solution to the short-term problems. For instance, if imparting training to the employees solves problems in the short-term, then companies schedule training programmes for the medium term. Similarly, if there are quality issues, then the medium-term response is to revise and improvise the company's quality control programme.

iii. Long-Term

In long-term strategic planning, companies endeavour to address and solve the problems once for all to reach their overall targets. This type of planning evaluates the competitive status of a company in its social, economic and political environment, and builds up strategies to achieve long-term goals.

1.5.1.2. Main objectives of production planning

A summary of main objectives of production planning is produced below:

- Steady flow of production
- Estimate the resources and ensure effective utilization of resources
- Ensures optimum inventory
- Minimize wastage of raw materials and improves productivity.
- Results in consumer satisfaction
- Reduces production costs

1.5.2. Material Requirement Planning (MRP)

MRP is a planning tool which benefits a company in the following ways:

- Reduced Inventory Levels and shortages
- Improved deliveries and productivity
- Simplified and Accurate Scheduling
- Reduced Purchasing Cost
- Reduced Manufacturing Cost
- Reduced Lead Times

- Improved Communication

MRP uses the following major inputs for arriving at the requirement planning decision:

- Master Production Schedule (MPS)
- Bill of Materials (BOM)
- Quantity on Hand (QOH)
- Part, component, raw material Lead Times
- Sales/ Purchase Order Quantities / Due Dates
- Scrap Rate
- Lot Sizing policies for All Parts
- Safety Stock Requirements

Below are certain terminologies used in MRP:

ABC analysis	ABC analysis is done to categorize items in stock into different groups based on their annual usage (based on Pareto analysis approach).
Actual cost	It is the total cost incurred on the job during its entire production process, which includes labour cost, material cost and associated overhead costs.
Available capacity	It is the capacity of the current bottleneck resource which decides the production in normal condition.
Backlog	The customer orders which have crossed the due date but not yet delivered are referred to as backlog.
Backward scheduling	It is the reverse approach to determine the date of release of the order by calculating the schedule back through routing from the actual delivery date.
Bill of materials	The constituents of the parent items that includes all the assemblies, sub-assemblies, components, parts and raw materials; along with their quantity.
Bottleneck	The resource which decides the overall output of a production system.
Buffer stock	It is the safety stock that is maintained in the inventory to take care of any sort of demand or supply fluctuations.
Business plan	A report which gives projections on current/ future income, products

	etc.
Critical resources	Manpower, machine etc which are critical for production.
Cumulative lead time	The total duration required to finish all the activities on the job.
Customer service level	The percentage of occasions when timely delivery to the customer is done.
Cycle time	Time taken to complete one manufacturing cycle.
Database	This is the data which is stored in a storage device.
Delivery performance	It is a measure of suppliers' capability of giving timely delivery.
Due date	Last date for completion of an operation.
Economic order quantity	EOQ is order quantity which results in the least cost when inventory carrying cost, ordering cost and set-up cost are taken into account wrt customer demand.
Effectivity date	The date from/till which an operation/component is valid in BOM or assembly process.
Exception reports	Reports generated when there is difference between actual value and expected value and the difference is more than the tolerance.
Expediting	Done to speed up the production activities or purchase orders placement to meet the production target, in order to keep supply in line with demand, which may be because of the priority or to prevent backlog.
Feedback	Inputs taken to monitor the performance of the process, which is done to correct or further improvement.
Finite capacity planning	It is the Capacity requirement planning that is done taking into account, the various capacity constraints that exist or may come during the operation.
First in, first out (FIFO)	It is a rule which says that the item which is received first should be delivered first. This is done to prevent the expiry of the items having shelf life.
First in, last out (FILO)	In this method, the items which are received in the beginning are issued last. For example, storage of sacks of cement where the first sack remains at the bottom while the new ones get stacked on top and are

	issued accordingly.
Forward scheduling	A scheduling process where the start date for an order is known and end date is calculated by analyzing the time taken between the first operation (the starting point) and the operation which is last.
Frozen zone	The time duration in the production schedule where no alterations are allowed in the master schedule.
Inventory	Raw material, WIP, stock, finished goods etc.
Lead-time	Time duration to complete an activity from the beginning to the end.
Multi skilled workers	Workers who can perform a variety of tasks.
On-hand inventory	The qty of an item that is not allocated to any other job and is readily available if required.
Overdue work	Work which has crossed the due date.
Overloaded capacity	Over utilization of the work centre.
Pareto analysis	ABC analysis done to categorize items in stock into different groups based on their annual usage.
Part master record	A master data containing various information of item like its description, lead time, order qty etc.
Pegged requirements	This is the requirement of a particular level of item which is calculated from the next level parent item (as given in the multi-level-BoM) or customer demand.
Capacity	Total volume of work load which can be handled.
Performance evaluation	Performance measurement.
Planning horizon	The master schedule extension duration in the future.
Product costing	Cost determination of a product.
Productivity	Output produced per unit input
Progress reporting	A Report which gives the current status of the job.
Queue	Waiting line at the work centre where the job is to be processed.
Queue time	The time duration for which the job waits in the queue.

Reprioritising	Priority revision based on latest information and requirement.
Routing	Detailed description of item manufacture, including operations pending to be performed, the sequence of the operations and the work centres at which the operations are performed etc.
Safety time	Material are supposed to be received few days before its need date.
Scheduled receipt	These are the items which are pending for receipt in a particular time period.
Scheduling	Assigning dates to the manufacturing products important steps.
Standard cost	Cost calculation based on Std. rates of materials, labour etc.
Standard time	It is the standard no.of hours required to set up a machine and run a part, assembly, batch or end product through it.
Start date	Date of initiation of an operation.
Stock out costs	Any cost which is associated with lost sales due to stock unavailability.
Storage costs	Is the sum total of all the costs which can be assigned to various elements of a warehouse.
Subcontracting	Providing Free-Issue-Material and outsourcing certain amount of work to another manufacturer.
Supplier performance	Metrics used to evaluate supplier performance over a period of time.
Time horizon	Time period under consideration.
Traceability	Production part, process, and material tracking by its lot or serial number.
Two-bin system	Inventory is carried in two bins in this type of order system. When the first bin gets empty, replenishment qty, equivalent to two bins is ordered.
Where-used list	A listing of every parent item that calls for a given component and the respective quantity required, from a BOM file.
Work in progress	Any job which is waiting for further processing in the shop floor.

1.6. Lot Sizing Problem (LSP)

The field of LSP attracted a lot of research due to its immediate impact on levels of inventory and hence on the inventory ordering/ set-up/ holding cost. The lot-sizing problem is in

principle concerned with finding order quantities that are able to minimize the total cost of a lot. The net requirements are grouped for a number of periods and the order quantity is decided in advance. Lot-for-lot is one of the popular techniques to identify lot quantity in the sense that whatever needed is ordered. Another method is ordering a fixed quantity every time disregarding the demand. Yet, another method is to cover the net requirements for a number of future periods, called fixed periods.

In practical scenario, a lot of industry, use a combination of the methods explained earlier.

Below section represents some of the critical characteristics which influence the quality of a lot sizing decision:

1.6.1. Lot Sizing Terminology

- **Number of resources:** The processing of the items can be done on single machine or multiple machines i.e. a single resource or multi resource model. In case of multi resource (parallel machines) models, timing, production levels and allotment of production lots on machines are required to be done which is a complicated task.
- **Number of levels:** A single level production system is a system in which final product is obtained directly in a single operation. There is no intermediate item in this case. The product demand is equal to the customer order or the forecast done. However, in a multi-level production system (which is a common occurrence), there is a parent-child relationship between the final product and the intermediate items (including raw materials). Since the input of an operation is actually the output of another operation, the demand at a level is dictated by the decision on lot size at the parents' level making this type of problem more difficult to solve.
- **Planning horizon discretization:** The lot sizing problem can be categorized into two types: big bucket or small bucket. In big bucket problems, the time period is long enough for producing multiple types of items, whereas in small bucket problems the time period is short and only one type of item is produced in each time period or bucket.

1.6.2 Lot Sizing Techniques in MRP

Some of the most commonly used traditional lot sizing techniques in MRP are given below:

i. Lot for Lot (LFL)

Period by period coverage of net requirement is considered as the lot size. Planned order quantity always equal to the net requirement.

ii. Fixed order Quantity (FOQ)

As per the FOQ rule the same order quantity is maintained each time as order issued. The order quantity may be taken as an integer multiple of average demand.

iii. Economic Order Quantity (EOQ)

The EOQ technique determines the lot size that minimizes annual inventory holding and ordering costs. The lot size is determined by using the formulae:

$$EOQ = \sqrt{\frac{2DC_O}{C_C}}$$

where

C_O= Ordering cost

C_C=carrying cost

D= annual demand

iv. Wagnor Whitin method (WW)

The WW model is the first formulation of a lot sizing problem for dynamic demand. This model assumes that demand is given by period and varies over time. This is initially developed for single item single level uncapacitated lot sizing problem. Wagnor and Whitin also developed an exact algorithm based on dynamic algorithm to solve these problems. Consequently, a large number of heuristics were developed with the idea of minimizing average setup costs and inventory cost over several periods.

v. Silver Meal heuristic (SMH)

In this method, entire planning horizon is broken down into sequence of time windows and order placement for item ‘i’ is done at beginning of each window.

Total cost for first time window ($TIC_{i,t}$) is given by

$TIC_{i,t} = K_1 + \sum_{p=1}^{t-1} H_P \sum_{q=p+1}^{t} d_q$

where

K=setup cost

H_p=inventory carrying cost

D_q= demand in that period

T=first time periods (1≤t≤B)

B=length of the planning horizon

The average cost per period

$$TIC_{(smh)i,t} = \frac{TIC_{i,t}}{t}$$

After calculating the average cost for t = 1, 2, 3,….etc, the process is carried out until a cost $TIC_{(smh)i,v}$ is found such that $TIC_{(smh)i,v} < TIC_{(smh)i,v+1}$ or if v = B. The same procedure is repeated until the entire planning horizon is covered.

vi. Least Unit Cost (LUC)

This technique attempts to bring the average cost/ unit to the lowest in each period of planning horizon. The same procedure which is used to calculate the total cost for the first period, and then the average is calculated as

$TIC_{luc} = TIC_{1,t} / \sum_{p=1}^{t} d_p$

After calculating the average cost for t=1,2,3…. etc, the process is carried out until a cost $TIC_{(luc)1,v}$ is found such that $TIC_{(luc)i}$, $v<TIC_{(luc)i,v-1}$ or if v=B. The same procedure is carried out until the entire planning horizon is covered.

vii. Add Drop heuristic (ADH)

This technique considers the capacity constraints which are possible in production. In this method, the total cost is calculated as

TIC = setup cost + holding cost + WIP cost + queue inventory carrying cost.

viii. Part Period Balancing (PPB)

If one part or unit is kept in the inventory for one period, it incurs a particular holding cost; if it is kept for two periods, it incurs twice the holding cost. One of the way to present ordering costs in terms of part period is to divide the ordering cost by the inventory holding costs per part per period.

BPSO Procedure

Initial population

In this method, a random particle population is created for the binary PSO algorithm for LSP. The dimension values are constructed in a random fashion. The values are binary i.e. 0 or 1 for each dimension of a particle with a probability of ½.

In particular,

X0id= 0 or 1

The values of the velocity are within a maximum and minimum value i.e. Vki (Velocity Value) = [Vmin,Vmax] = [-4,4], where Vmin = -Vmax. The particle velocity in dth dimension is given by

$$v_{id}^{0} = V_{\min} + \left(V_{\max} - V_{\min}\right) * rand()$$

For a particle 'i', the enhancement of problem space of the local search exploration is done by this limit. The number of dimensions is half the Population size. The total ordering and holding cost is to be minimized, which is the aim of the formulation of the LSP. The fitness function value for the particle 'i' is given below:

$$f\left(X_i^k\right) = \sum_{j=1}^{d} \left(Ax_{ij}^k + cI_{ij}^k\right).$$

Finding new solutions

Because of the binary version employed in the study for PSO algorithm, two functions shall be used for new solution generation, a sigmoid function which forces generation of real values, values (0,1) and a piece wise linear function which forces velocity values not to go beyond the prescribed maximum and minimum values.

The piece-wise function is given below:

$$h\left(v_{id}^{k}\right)=\begin{cases}V_{\max}, & if \quad v_{id}^{k}>V_{\max}\\ v_{id}^{k}, & if \quad \left|v_{id}^{k}\right|\le V_{\max}\\ V_{\min}, & if \quad v_{id}^{k}<V_{\min}\end{cases}$$

After applying above function, the sigmoid function scales the velocity between 0 and 1 (used for converting them to the binary values) i.e.

$$sigmoid\left(v_{id}^{k}\right)=\frac{1}{1+e^{-v_{id}^{k}}}.$$

So, new solutions are found by updating the velocity and dimension respectively. First, compute the change in the velocity V^{k}_{id} such that

$$\Delta v_{id}^{k-1}=c_1r_1\left(pb_{id}^{k-1}-x_{id}^{k-1}\right)+c_2r_2\left(gb_{d}^{k-1}-x_{id}^{k-1}\right)$$

After computing the change in velocity, the velocity V_{id} is updated by utilizing the piece-wise linear function:

$$v_{id}^{k}=h\left(v_{id}^{k-1}+\Delta v_{id}^{k-1}\right).$$

Finally, the dimension d of the particle is updated:

$$x_{id}^{k}=\begin{cases}1, & if \quad U(0,1)<sigmoid\left(v_{id}^{k}\right)\\ 0, & otherwise\end{cases}.$$

1.6.3. Strengths and Weakness

i. Strengths

The rise in popularity of PSO is because of its simplicity. PSO is a straightforward and computationally simple technique. A number of researches have shown that PSO performs better than other algorithms on a variety of problems. Also, PSO has been shown to be more competitive on others on a number of problems. PSO is a new technique, and a lot of researchers have done work to improve the original PSO. As such, PSO has a great potential. e.g. many successful evolutionary computation (EC) techniques and ideas may be integrated

to improve PSOs, owing to its similarity to EC methods. Like many EC algorithms, PSO has a number of parameters to adjust which is beneficial for implementing adaptive systems. This also shows the extensibility of PSO to other specifically designed algorithms although it may not perform as well as those algorithms. Although, tuning parameters for solving a particular problem or a range of problems can be time-consuming and non-trivial. Compared with EC methods, PSO does not have as many parameters to tune in order to get acceptable performance. In addition, Hu and Eberthart suggest that PSO is applicable for both constrained and unconstrained problems even without pre-transforming the constraints and the objectives of a problem.

ii. Weaknesses

Researchers have found a lot of issues which inhibit the generic PSOs in providing an effective solution for certain types of problems. For example, although PSO has the ability to converge quickly, it tends to wander and slow down as it approaches an optimum. Owing to the premature convergence, it gets stuck quite easily and hence, it cannot explore wide enough. This can be problematic for solving multimodal problems where the problems have multiple optimal solutions. Particularly if many of those Optima are only local rather than global, particles may get trapped at local optima. In addition, while there are not many parameters to control and as mentioned previously, these parameters open up a potential for developing adaptive PSO systems. Some suggested values and experimental settings are still at trial-and-error stage, and it can be non-trivial to find the right settings for individual problems.

CHAPTER 2

LITERATURE SURVEY

2.1. Introduction

This chapter presents literature review of the work carried out by the researchers on Traditional and Non-Traditional Techniques for solving scheduling and lot sizing problems. The literature on Scheduling theory/ practice is abundantly available, whereas literature on lot sizing is available in limited numbers. The review on scheduling is presented in 2.2, 2.3, 2.4, 2.5 sections and review on inventory and lot sizing is presented in 2.6, 2.7 sections.

2.2. Scheduling

Job Shop scheduling is most popular problem in Operations Research literature. There is widespread belief that the JSSPs are very difficult to solve. The scheduling of Job Shop has been widely studied as deterministic scheduling model and the solutions of scheduling problems have been a test bed for those who have developed/ are developing new techniques for scheduling of JSSPs.

There is a mixed view, on who was the first, to propose JSSP in its current form. The disjunctive graph representation was first proposed by Roy and Sussmann in 1964 and based on this, Balas first applied an enumerative approach in 1969. However, it is known that a lot of work have been done in JSS even before. Johnson proposed flow shop algorithm to the job shop in 1954, which was later generalized by Jackson in 1956. In 1955, Akers and Friedman represented processing sequences by applying Boolean Algebra approach. Priority dispatch rule template was proposed by Giffler and Thompson in 1960. These works cite earlier references also. For example, Akers and Friedman represented processing sequences by applying Boolean Algebra approach in 1955 (a working paper by Salvesen and Anderson).

The work done on JSSP dealt with a problem consisting of n number of jobs, m number of machines. These problems had precedence relationships also. In these problems, each job was processed in varying order on various machines. The objective was to complete all the tasks in least possible time. Tables 2.1 and 2.2 give a summary of the main researchers, who used conventional and non-conventional concepts for solving JSSPs.

S.No	Method	Author1	Author2
Problem Representation			
Disjunctive Graph		Roy and Sussmann (1964) White and Rogers(1990) Radermacher(1985)	Bartusch et.al. (1988) Sussman (1972) Sotskov (1997)
Problem Classification NP Classification		Conway et.al.(1977) Lenstrae et.al.(1977) Cook(1971) Garey and Johnson(1979) Sotskov and Shakhlevich (1995)	Graham et.al. (1979) Brucker et.al.(1996a,b) Garey et.al. (1976) Lenstra and Rinnooy Kan (1979) Timkovsky (1995)
Gantt Chart		Gantt(1919) Porter(1961)	Clark(1922)
Optimization Algorithms		Lawler et.al.1993)	
(I)Efficient Methods Solvable In Polynomial Time		Hardgrave(1963) Johnson(1954) Jackson (1956) Brucker(1988,1994) Williamson et.al. (1997) Hefetz and Adiri(1982) Kravchenko and Sotskov(1996)	Nemhauser (1963) Akers(1956) Szware(1960) Sotskov(1985) KubaikandTimkovsky,1996 Kravchenko(1995) Brucker et.al.(1997a,b)
(II)Enumerative Algorithms		Lenstra(1976)	Rinnooy Kan(1976)
1. Mathematical Formulations		Lawler(1983)	Blazewicz et.al. (1991)
(i)	Linear Programming (i)Mixed (ii)Integer	Bowman(1959) Vanden Akker(1994) Balas (1965)	Dyer and Wolsey(1990) Wagner(1959)
(ii)	Decomposition Techniques	Chu et.al. (1992) Ashour(1967)	Kruger et.al. (1995) Kanzedal(1983)

(iii)	Lagrangian Relaxation (Augmented)	Hoitomt et.al. (1993) Fisher(1973a,b,1976) Hoogeveen(1995)	Vande DeVelde(1995) Fisher et.al. (1995) Vande DeVelde(1991)
(v)	Miscellaneous	Blass(1979,1985)	Applegate and Cook(1991)
(iv)	Duality constraint	Glover(1968,1975,1977)	Fisher et.al. (1983)
2. Branch and Bound(BB)		Lageweg et.al.(1997)	Pinson(1995)
(i)	DisjunctiveGraph	Balas(1969) Mc.Mahon and florian(1975) Florian.et.al.(1971) Brooks ND White(1965) Ashour and Hiremath(1973) Bratley.et.al.(1973) Alessandro Mascis(2002) Applegate and Cook(1991) Brucker.et.al.(1994)	Dario Pacciarelli(2002) Greenberg(1968) Perregaard and Clausen(1995) Charlton and Death(1970a,b) Nabeshima (1971) Barker and McMahon Ashour and Parker(1973) Ashour et.al. (1974) (1985) Carlier and Pinson(1989,90,94)
(ii)	Time Orientation	Martin(1996)	

Table 2.1 Methodologies to solve JSSP by Conventional Techniques

2.3. Traditional Scheduling Methods

The most popular method of solution representation is the Gantt chart referred in Gantt (1919), Clark (1922) and Porter (1968). However, Blazewicz et al. (1996) indicates that Roy and Sussmann, (1964) disjunctive graph model is now extensively existing. In 1990, White and Rogers studied the limitations of these models. It is very difficult to directly model parallel and cyclical process flows, therefore, application of this model to actual scenarios in industry becomes difficult. The successful extension of disjunctive graph model for representation of specific cases in various areas pertaining to manufacturing, maintenance, material handling was done by White and Rogers.

In 1993, Mac Carthy and Liu took non-basic models also in account and proposed a two methods combination. Johnson, in 1954 has developed optimization method called Johnson algorithm for a two-machine flow-shop. Few other efficient proposals also exist for the JSSP of the 2-jobs on m-machines and n-jobs on 2-machines (Jackson, 1956). During the years in 1950s, a number of problems were solved by applying some raw but effective techniques and later on, these became the foundations for further advancement in classical scheduling. If 1950s was the decade for application of raw techniques, 1960s was the decade when the interest of the researchers shifted to applying enumerative algorithms adopting an elaborate mathematical model (which were sophisticated also). Rinnooy Kan, (1976) attempted "a natural way to attack scheduling problems using mathematical programming models". Although, these work failed to meet expectations and most of these strategies were not able to give feasible solution to many problems. Hence, their practical usage is very limited and used only in lower bound calculation.

Branch and Bound (B&B) is one of the important enumerative strategies. In B&B, the solution space is generated in the form of a tree. Branch and Bound gave procedures and rules so that a large part of the tree was allowed to be removed from search. Branch and Bound was one of the popular techniques of JSSP for a considerable time. This method was found to be very suitable, however for a case like $N < 250$ (which requires enormous amount of computing), it is not possible to apply this method. In addition, the way these methods perform, depends on upper bound of the solution (Lawler et al., 1993).

From 1970s to mid-1980s, the focus was to justify the complexity of JSSPs. Lot of work, beginning with the work; Cook (1971) to Lawler et al. (1982) showed that JSSP is highly unmanageable and few special situations could be solved in polynomial time. This is the reason why, era before 1970 is referred to as BC (Before Complexity) by Parker (1995). The era after 1970 is referred to as AD (Advanced Difficulty). In 1979, Garey and Johnson referred to 320 NP-Hard problems. Therefore, approximation methods became an alternative since there were limitations to the exact enumeration techniques.

It has already been explained that these techniques guarantee speed at the cost of an optimal solution, hence, these techniques can be utilized for solving larger problems. Priority dispatch rules (pdrs) are the approximation algorithms which were the earliest. In this method, priorities are assigned to operations. The operation having the highest priority is chosen first. Their implementation is quite easy and computational burden also is low. The work (Panwalkar and Iskander, 1977) contains a list of such rules. Further, the research carried out in these field shows that the best techniques are those which are linear or randomized or combination of a number of priority rules. Fisher and Thompson (1963), Panwalkar and Iskinder (1977) and Lawrence (1984), Fuzzy Logic (Grabot and Geneste, 1994) and Genetic Local Search (Dorndorf and Pesch, 1995) are being widely used because of their innovative approaches. Nevertheless, these works draw attention to the nature of pdrs which has high dependence on nature of the problems. This is clear from the case of makespan minimization in which no single rule demonstrates superiority.

S.No	Method	Author1	Author2
Approximation Algorithms		Fisherand Rinnooy Kan(1988)	Blazewicz et.al.(1996)
(I) Constructive Methods			
(1) Priority Dispatch Rules		Gere(1966) Jackson(1955) Giffer and Thomson(1960) Crowston et.al.(1963) Panwalkar and Iskander(1977) Lawrence(1984)	Smith(1956) Fisher and Thompson(1963) Jeremiah et.al.(1964) Moore(1968) Haupt (1989) Blackstone et.al.(1982)
(2) Insertion Algorithms		Winkler and Werner (1995)	

(i)	Bottleneck based heuristics	Alvehus(1997)	
(ii)	Shifting Bottleneck Procedure(SBP)	Demirkol et.al.(1997) Adams et.al.(1988)	Holtsclaw and Uzsoy(1996) Cook and Applegate (1991)
(II) Iterative Methods			
(1) Artificial intelligence		Greenberg and Glover (1989)	
i)	Constrained Satisfaction(CSPs)	Eerschler et.al.(1976) Nuijten & Aarts(1994,96) Harvey and Ginsberg(1995) Baptiste and Le Pape(1995) Sadeh(1991) Pesch & Tetzlaff(1996)	Fox(1987) Caseau and Laburthe(1994,95) Harvey(1995) Baptiste et.al.(1995) Sadeh and Fox(1996) Nuijten and Le Pape(1998)
(2) Neural Networks		Wang and Brunn(1995)	Jain and Meerun(1998)
(i)	Hopfield Networks	Foo et.al.(1994,1995) Van Hulle(1991) Willems and Rooda(1994) Foo and Takefuji(1988a-c)	Satake et.al.(1990,91) Zhou et.al..(1990,91) Sabuncuoglu and Gurgun(1996) Loand Bravian (1993)
(ii)	Back Error Propagation (BEP)	Cedimoglu(1993) Daglietal.(1991) Kimetal.(1995)	Sim et.al.(1994) Watanabe et.al.(1993) Sittisathanchai & Dagli (1995)
(3) Expert Systems		wink and Biegel (1989) Alexander(1987) Shakhlevich et.al.(1996)	Chen and Kusiak (1988) Sotskov(1996) Charalambous and Hindi(1991)
(III) Local Search Methods		Evans(1987) Vaessens et.al.(1995,96) Mattfeld et.al.(2000)	Vaessens(1995) Arts and Lenstra(1997)
(1) Problem Space Methods			
(i)	Problem & Heuristic Space	Storer et.al.(1992,95)	

(ii)	GRASP	Resende(1997)	
(2) Genetic Local Search		Aarts et.al.(1991,1994) Della Croce et.al.(1994) Mattfeld(1996) Imen Essafi, Yazid Mati,	Pesch(1993) Stéphane Dauzère- Pérès(2008) Yamada and Nakano(1995b,1996b,c)
(3) Ant Optimization		Colorni et.al.(1995,96) Colorni, M. Dorigoet Goss, S. Aron JL.	V.Maniezzo(1991) Deneubourget J. M. Pasteels
(4) Reinsertion Methods		Werner and Winker (1995)	
(1) Threshold Algorithm		Aarts et.al. (1991 and 94)	
(i)	Threshold Improvement	Aarts et.al. (1991,94)	Storer et.al. (1992)
(ii)	Simulated Annealing	Aarts et.al. (1991,94) Matsuo et.al.(1988) Sadeh and Nakakuki (1996	Yamada et.al.(1994) Yan Laarhoven et.al. (1988,92) Nakano and Yamada
(2) Threshold Acceptance		Aartsetal.(1991,1994)	
(i)	Large step Optimization	Lourenco(1993,1995) Brucker et.al.(1996a,1997a)	Lourenco & Zwijnenburg(1996)
(2) Tabu Search		Taillard(1989,1994) Barnes and Chambers (1995) Nowioki and Smutnioki (1996) Thomson (1997)	Trubian and Dell'Amico (1993) Sun et.al. (1995) Marino Widmer (1996)
(IV) Evolutionary Algorithms			

(1) Genetic Algorithm		Davis(1985) Della Croce et.al. (1995) Nakano and Yamada (1991) Yamada and Nakano (1992) Rossetal.(1993) Normanand Bean(1995)	Falkenauer and Bouffouix(1991) Tamaki and Nishikawa(1992) Davidor et.al. (1993) Mattfeld et.al. (1994) Kobayashi et.al. (1996)
(2) Particle Swarm Optimization		D.Y. Sha(2006) Tsung- Lieh Lin, Shi-Jinn Horng, Tzong-Wann Kao,Yuan-Hsin Guohui Zhang(2009) Qun Niu, Bin Jiao,(2008) Deming Lei(2008) Weijun Xia(2005) D.Y.Sha (2009) Kun FAN(2007) Massimo Paolucci (2009) M. Rabbani, M. Aramoon	G.Baharian Khoshkhou(2009) Chen,Ray- Shine Run, Rong-Jian Chen,Jui- Lin Lai,I-Hong Kuo Xinyu Shao, Peigen Li, LiangGao(2009) Davide Anghinolfi(2009) Xingsheng Gu(2008) Cheng-Yu Hsu(2006) Zhiming Wu (2005)
(3) Memetic Algorithm		Liang Sun and Jin-hui Yang (2009) Heow Pueh Lee(2009) Lacomme, Nikolay Tchernev Nima Safaei, Farrokh Sassani Mohsen	YunQian(2009) Jalil Layegh, Fariborz Jolai,(2009) Yan-chunLiang(2009) Anthony Caumond, Philippe(2008) Reza
(5) Immune Algorithm		Bagheri and Zandieh(2009)	Mahdavi and Yazdani(2009)
(6) Hybrid Algorithms			
(i)	Hybrid PSO	Dongyun Wang(2008) Peng-YengYin, Shiuh-Sheng (2007)	Liping Liu(2008) Yu,Pei-Pei Wang,Yi-TeWang (2007) Dušan Teodorović(2008)
(ii)	Hybrid GA	Waiman Cheung (2009) Mitsuo Gen and Jie Gao, (2007)	Xiaohui Zhao and Linyan Sun (2007) Lawrence C.Leung(2009)

(7) Invasive Weed Optimization	Hymavathi. M(2012)	C.S.P.Rao (2012)
(8) Bacterial Foraging Optimization	Chunguo Wo Narender S(2012)	Jingqing Jiang Amudha T(2012)
(9) Hybrid Bacterial Foraging Optimization	Shivakumar BL (2012)	AmudhaT(2012)

Table 2.2 Methodologies to solve the JSSP by Unconventional Techniques

2.4. Non-Traditional Scheduling Methods

In the late 1980s, researchers realized the difficulties in solving JSSPs. This resulted in shifting of main focus of the research from emphasizing the NP Hard status of JSSP/ creation of optimization techniques to applying approximation methods for solving these problems.

The researchers felt the need of more suitable techniques (of applying a better improved perspective) because of the general imperfections found in pdrs. Fisher and Rinnooy Kan (1988) initiated work on these applications. Further, the survey done by White and Rodammer (1988) emphasized the flexibility and power of such heuristic approaches when compared with optimization techniques. Rodammer and White also highlighted that procedures of innovative heuristics search were legitimate tools for more real application scheduling problems besides JSSP.

The appearance of heuristic strategies that are inspired by intelligent problem solving and natural phenomena for solving difficult problems like JSSP, was indicated by Glover and Greenberg (1989). Approximation techniques were the major focal point of research during the boom period, but some of the efforts were still being targeted at optimization methods.

For example, Carlier and Pin- son (1989) proposed a Branch and Bound method that was able to prove the optimal solution to FT 10 for the first time. It is to be noted that for over 25 years, FT10 problem was eluding researchers. Pinson and Carlier (1990), Applegate et.al.(1991) have solved instances larger than FT10 using Branch and Bound techniques.

Prior to 1988, priority dispatch rule (Lawrence, 1984) was the single available technique for solving instances with more than 100 operations. The work of Fox (1987), Takefuji and Foo (1988a-c), Zhou et al., 1990 & 1991, Sadeh (1991) in dealing with these problems can be referred in this regard.

Some of the innovative algorithms conceived in the period during the late 1980s (including earlier years of 1990s) to solve JSSP are enumerated below:

(a) Artificial Immune Systems (AIS) (J.D. Farmer et. al,1986), (b) Tabu Search (TS) (Taillard, 1989; Glover, 1989, 1990); (c) Large Step method of Optimization (Martin et al., 1989); (d) Simulated Annealing (SA) (Matsuo et al., 1988; Van Laarhoven et al., 1988; Aarts et al., 1991); (e) Genetic Algorithms (GAs) (Bouffouix and Falkenauer, 1991; Nakano and Yamada, 1991), (f) Genetic Local Search (GLS) (Moscato, 1989; Aarts et al., 1991), (g) Ant colony Optimization (ACO) (Dorigo et al, 1992), (h) Particle Swarm optimization (PSO) (Kennady & Eberhart, 1995), (i) Hybrid methods, (j) Memtic algorithms, (k) Invasive Weed Optimization (IWO) (Meharbian,), (l) Bacterial Foraging Optimization (BFO) (Passino), (m) Harmony Search (HS), (Geem et al, 2001) which have been described later in this thesis. The work of Matsuo et al., 1988; Van Laarhoven et al., 1988, were derived from Balas (1969), and Gra- bowskiet al. (1988) and this laid the basis for powerful JSSP neighborhood structures. These utilize local search methods using the notion of blocks and critical operations.

Adams et al., 1988 proposed SBP technique which has the biggest impact on approximation methods. This method started the boom period and became the first heuristic which was able to solve FT10. This algorithm was able to get better results because of its algorithm and software which were well developed for solving the one-machine scheduling problem. Demirkol et al., 1997 proposed simple tasks of Sub problem identification, bottleneck selection, sub problem solution, and schedule re-optimization characterize SBP.

The strategy which is actually used involves breaking down the problem to the scenario equivalent to m one machine problems. Once the problem is broken to this level, each of the sub problems are solved one at a time. Each solution obtained from the solution of the sub problems are compared with the others. A ranking is done and the machines are arranged in that order. The bottleneck machine is the machine which has the largest lower bound. The

next step is to first sequence the bottleneck machine while ignoring the still existing un-sequenced machines. The machines which have been already scheduled are held fixed. Every time scheduling is done for the bottleneck machine, the previously sequenced machine (which can be improved) is locally re-optimized by solving the problem of one machine again.

Carlier (1982) has solved one machine problem using iterative method. The main contribution of this approach is the way the one machine relaxation is used to decide the order of scheduling of machines. Holtsclaw and Uzsoy (1996) and Demirkol et al. (1997) gave a computational analysis (which was comprehensive) and Alvehus (1997) gave a thorough review of this method. Nevertheless, the basic issue of SBP is the difficulty in performing re-optimisation and the infeasible solutions generation. The substantial differences were noted by Lasserre and Dau-zere-Peres (1993), Adams et al. (1988) and Balas et al. (1995), when applying four reoptimisation cycles (rather than three). The work (Vazacopoulos and Balas, 1998) amal-gamates led variable search with SBP producing one of the best JSSP approaches is available. Even though Balas and Vazacopoulos (1998) have suggested a number of detailed reoptimisation schemes, there is no strategy, as of now, to show how this should be done. Additionally, method is not available for deciding the sub problem size or to decide the machine(s) which are required to be fixed and also to solve the problems (with more structured job routings), SBP will need to be modified. The work of Caseau and Laburthe, 1995 Vazacopoulos and Balas, 1998; Na-kano and Yamada, 1996a; Vaessens, 1996 in which SBP procedure is incorporated for improving the upper/ lower bounds of several hard problems can be referred for further details.

Morton (1990), for example, took SBP to the scheduling of projects and applied a number of different performance criteria. The work of Ovacik and Uzsoy (1992, 1996), Ramudhin and Marier (1996), Ivens and Lambrecht (1996) and Balas et al. (1998) in adoption of this technique to semi-conductor testing facility, assembly and parallel machines and application to JSSP with deadlines, are noteworthy.

It has already been stated that the boom period created and formalized a number of approximation methods which are now being described. Two phases which constitute the insertion algorithm of Winkler and Werner (1995) are:

(i) In the first one a constructive strategy is applied in which an operation is positioned in the schedule in order that the length of the longest path passing through it is minimized, and

(ii) a reinsertion strategy is applied for the improvement of the initial solution iteratively.

The approach adopted in Beam search resembles the list strategy of tabu search (Laguna and Glover, 1997) in the sense that it also allows the search of a limited number of parallel solutions. As in BB approach, the choice of best lower node is based on lower bound. A filtered beam search method was demonstrated by Bayiz and Sabuncuoglu (1997) and the obtained results show that the technique used by them provides significant improvement as compared to the results.

Instances of iterative approximation methods are available which utilize a lot of concepts used in Branch and Bound algorithm; one of the examples is Constraint Satisfaction technique. A lot of these methods find it difficult to represent the constraints and require backtracking which is unnecessary. This results in reaching (or converging) to dead end states. In general, poor results (requiring excessive computation efforts) have been achieved by these methods (Caseau and La- burthe, 1995; Pesch and Tetzlaff,1996; Nuijten and Le Pape, 1998), in spite of the high-spirited boom period witnessed. Similarly, if considered the cases of successful application of ANN (Artificial Neural Network) to solve the JSSP in single machine environment, only the work of Sabuncuoglu and Gurgun (1996) can be referred to as successful application to the benchmark problems. Therefore, same conclusion can be drawn to Artificial Neural networks techniques also.

Other iterative methods which are noteworthy are problem-space methods which generate a lot of different starting solutions based on fast problem-specific constructive methods and improved further by local search. Greedy random adaptive search procedure (GRASP) (Resende, 1997) and Storer 1992, 1995 belong to this class of technique.

The most commonly accepted or prevalent method is threshold algorithm, in which the difference between the cost of present solution and the neighbor is lower than the threshold. The threshold algorithm in this situation chose a configuration which is new. Simulated annealing (SA) iterative improvement and threshold acceptance is some of the members of

this group of algorithms. Out of these two techniques, SA can be regarded as the most popular technique. The reasons of popularity of SA with iterative improvement have been applied extensively to JSSP.

SA is a local search technique with random orientation, the basic concept of which, has been derived from statistical physics. The technique imitates the annealing process when done on a metal (at elevated temperatures) which gradually cools to a state of minimum energy (or a stable state). The SA, being a generic technique, could not achieve good solutions to JSSP problems quickly. Therefore, researchers were looking at combining other methods with SA in order that the results are improved and required computing time is reduced. The works of Yamada and Nakano, 1995a, 1996a and GA (Kolonko, to appear) can be regarded for this purpose, although the computation efforts are still excessive.

Search techniques based on to the highest level compatible with the environment (constraints of the problem) are the basis of Genetic Algorithms (GAs). GAs are not able to successfully represent JSSP and crossover operators cannot generate feasible schedules without degradation of their efficiency. Additionally, many of the GA methods are not able of converging to an optimal solution. These deficiencies laid the basis for initiation of work on Genetic Local Search (GLS) and which is also referred to as population based local search or memetic search (Grefenstette, 1987; Moscato, 1989; Ulder et al., 1990, 1991). The recent work (Mattfeld, 1996; Yamada and Nakano, 1996b, c) are noteworthy to be referred in this regard. An example of a bilevel local search technique such as Large Step Optimization is GLS. This optimization encompasses a dual phase optimization method which consists of a large optimized transition known as “large step” and then a local search method known as “small step”. This was developed by Martin et al. (1989, 1992). Iterative improvement or SA is used to perform small steps most commonly. In large steps, problem specific techniques are applied for local minima to be transcended at low temperatures. It is, in comparison, a recent technique and has only been applied to JSSP by Lourenco (1993, 1995) and Lourenco and Zwijnenburg (1996). The analyses which have been done, although limited, indicate that this technique requires very high computation and provides good results than that of the local search method. A similar type of bi-level strategy was also applied by Brucker et al. (1996a, 1997a), to a wider domain of scheduling problem and this strategy has given some encouraging results. Tabu Search (TS) by Glover (1977, 1986, 1989, 1990) is a successful iterative approximation approach for JSSP. The technique of Nowicki and Smutnicki (1996),

at present, is regarded as one among the most powerful TS approaches that help in achieving good solutions rapidly. These deficiencies initiated the work on Swarm Intelligence Methods like Particle Swarm Optimization (PSO), Bacterial Foraging Optimization (BFO), Ant Colony Optimization (ACO), Bee Colony Optimization (BCO), Firefly Algorithm (FA) and Cockoo Search (CS) and Invasive Weed Optimization (IWO) and Harmony Search (HS), and which were also referred to as population based heuristics for global optimization.

2.5. Literature review on Particle Swarm Optimization

A swarm is a collection of organisms or agents which interact with one another. The term swarm intelligence (SI) was first used in reference to Beni and Wang's cellular robotic systems in the late 1980s. In their systems, a group of simple robots interact with their neighbor robots via communication. Later (in the early 90s), swarm intelligence studies were inspired by social insects, birds, fish and human cognition. In the recent years, swarm modeling has proved to be a new strategy for the solution of both constrained and unconstrained optimization problems.

Kennedy J, Eberhart RC (1995) are the first persons who introduced this PSO. As already explained in the previous chapter, PSO is an evolutionary computation technique which copies the behavior and information exchange process observed in the flock of bird flying. PSO possess high search efficiency since it combines local search (by self-experience) and global search (by neighboring experience).

Chandrasekaran.S, Ponnambalam.S.G, Suresh. R. K, Vijayakumar.N (2006) have studied the scheduling problem in flow shops for minimization of makespan, total flow time and total completion time. A simple version of Particle Swarm Optimization Algorithm (PSO) has been used for the solution of the set of Taillard benchmark flow shop scheduling problems.

Rahimi – Vahed. S.M.Mirghorbani (2007) used a bi-criteria permutation in flow shop scheduling problem, where, simultaneous minimization of weighted means of both the completion time and tardiness is attempted. Zhixiong Liu.(2007) proposed operation based particle representation. Mapping between the particle and the scheduling solution is done by connecting the operation sequence of jobs with the particle position sequence. The particle representation ensures the scheduling solutions are feasible and best candidate of the particle swarm optimization algorithm.

Simulated annealing (SA) employs certain probability to avoid being trapped in a local optimum. SA has shown its effectiveness in varied situations which include scheduling and sequencing. By a reasonable hybridization of these two methodologies, they develop a hybrid method for the multi-objective flexible job-shop scheduling problem (FJSP) which is easily implementable.

Sha, Cheng-yu Hsu.(2006) proposed a hybrid particle swarm optimization (HPSO) for the job shop problem (JSP). Since the JSP solution space is discrete, modified the particle position representation, movement, and velocity of the particle and so that the PSO suits the JSP in a better way. Giffler and Thompson's heuristic has been utilized for decoding a particle position into a schedule. They also applied TS for improvement of the quality of the solution. Davide Anghinolfi, Antonio Boccalatte, Alberto Grosso, Massimo Paolucci, Andrea Passadore, Christian Vecchiola. (2009) presented a multi-agent search approach to for a single machine total weighted tardiness problem with of sequence-dependent set up times.

I-Ling Lin.(2005) presented a PSO paradigm for the design and evaluation of a variety of new constraint-solving algorithms. Constraint satisfaction problems (CSPs) are applied for getting the solution of a number of practical problems but they are generally NP-hard, therefore development of new methods is a major challenge in research. PSO is a relatively new approach to AI problem solving and its application to CSP has just started. The algorithms that combine zig zagging particles and repair-based CSP-solving methods perform best among the algorithms studied.

Ajith Abraham, Hongbo Liu, Tae-Gyu Chang. (2009) introduced a hybrid metaheuristic called the Variable Neighborhood Particle Swarm Optimization (VNPSO) algorithm. The proposed VNPSO algorithm is utilized to solve the multi-objective Flexible Job-shop Scheduling Problems (FJSP).

Quan-KePan, M.Fatih Tasgetiren, and Yun-Chia Liang.(2007) presented, a discrete particle swarm optimization (DPSO) algorithm for the solution of the permutation flow shop sequencing problem. A new crossover operator is presented. The PTL crossover operator is able to generate a pair of different offspring even from the two identical parents. Additionally, the DPSO algorithm is hybridized with a simple local search algorithm based on an insert neighborhood for further improvement of the quality of the solution.

2.6. Literature Review on Lot Sizing Problems

A multi item lot sizing problem of N items can be reduced to N uncapacitated single item by imposing no capacity restrictions. Each of these single item problems may be solvable in polynomial time. Since the first work of Manne, the capacity constraint has been a major challenge. The problems in these cases have been called: The Capacitated Lot Sizing Problem (CLSP). These problems, whether they are Multi item or single item cases, are NP hard. The CLSP with multiple items has attracted a lot of research attention in the literature.

The methods of solution of the problems can be categorized in three major heads based upon the literature. These major heads are (i) Exact methods, (ii) Common sense or specialized heuristics and (iii) Mathematical programming based heuristic. Some of the heuristic Techniques used for Lot Sizing problem.

- Exact Methods
- Common Sense and improvement Heuristics
- Mathematical Programming Heuristics
- Relaxation Heuristic
- Branch and Bound based heuristic
- Set Partitioning and Column Generation method

Lot sizing problems are mainly divided into 2 types like (a) single level lot sizing (SLLS), and (b) multi-level lot sizing (MLLS). The types (variety) of final products (Finished Goods, FGs) in a manufacturing/ production set-up is another critical characteristic that affects the complexity of production planning problems. These conditions make the modeling of the problem more difficult. Based on the types of FGs, two principle types of production system emerge: (i) One type of FG is produced which utilizes single item production planning, (ii) Multiple FGs are produced which utilizes multi item production planning. It is obvious that the complications in multi item production planning problems are far higher than the single item production planning problems. Further, these production planning decisions are affected by the capacity/ resource restrictions which may be capacitated (capacity constraints are explicitly stated) or uncapacitated (no restrictions in capacity). It may be noted that capacity/ resource comprises of manpower, machines, equipment, budget etc. It is evident that planning for multi item becomes more complex in capacitated situation. The subject of single level lot

sizing with variants has been addressed by several methods in the literature. A number of heuristic methods have been devised for the solution of LSP, but most of them are applicable for small instances. Very few approaches are implemented for MPMLLS problems. Some of those techniques are which are taken from literature are presented here.

2.7. Literature Review on Soft Computing Techniques

In 1958, Wagner and Whitin (2004) introduced the SLLS model and developed a Wagner and Whitin algorithm based on dynamic programming. After that, Silver and Meal (1973) proposed the concept of minimization of average setup and inventory costs on time horizon. Mc Knew and Coleman (1991) proposed a part period algorithm for minimizing setup and holding cost over different periods.

i) Genetic Algorithm Literature

In the year 1999 Hernández, W. and G. Süer, presented a GA approach for simple LSP. The LSP is defined as finding the order quantities for a single item (with single level BoM) case in which shortages are not allowed, and the capacity is infinite (uncapacitated). Experiments were conducted to evaluate how different aspects of the genetic algorithm affect the results. Particularly, it was observed that how scaling was having the biggest impact. After that, in 2001 N. Dellaert, J. Jeunet", N. Jonard, presented genetic algorithm approach for general multi-level uncapacitated LSP with time-varying costs. In this research, they used product cross over, periodic cross over, inversion, single bit mutation, multi bit mutation etc. After this work, in 2002, Jinxing Xie and Jiefang Dong used GA which addresses the issue of calculating the production lot sizes for various items over a given finite planning horizon. In 2006, Lotfi Gaafar applied genetic algorithms to dynamic lot sizing with batch ordering.

ii) Hybrid Genetic Algorithm Literature

In 2012, Yuli Zhang, Shiji Song, Heming Zhang, Cheng Wu, Wenjun Yin Studied a multi-item inventory system (with two stages) in which the demand was stochastic at stage 1. The replenishment of inventory for the nodes present at stage 1 was being done from stage 2. Monte Carlo method was used for the simulation of the actual demand and thus for the approximation of the long-run average cost. The extensively utilized installation and echelon policy has been compared by numerical experiments and the results depict that in case of

increase in the variance of stochastic demand, the echelon policy performs better than installation policy. Hence, the search capacity of HGA is greatly enhanced by the proposed heuristic search technique.

iii) Heuristic Algorithm Based on Segmentation

In 2007, Ikou KAKU, Zhaoshi LI and Chunhui XU, used a Heuristic Algorithm Based on Segmentation for Solving Large Multilevel Lot-sizing Problems

iv) Randomized multi-level lot-sizing heuristic technique

In 2009, N.P.Dellaert , J. Jeunet, used a Randomized multi-level lot-sizing heuristics for Production, Manufacturing and Logistics for general product structures.

v) A MAX-MIN Ant System

In 2005, Rapeepan Pitakaso, Christian Almeder, Karl F. Doerner, Richard F. Hartl used A MAX-MIN Ant System for solution to Unconstrained Multi-Level Lot-Sizing Problems. An ant-based algorithm is presented for the solution of unconstrained multi-level LSP known as ant system for multi-level lot-sizing algorithm (ASMLLS). A hybrid approach is applied in which Ant Colony Optimization is utilized for finding a good lot sizing sequence and in this a modified Wegner-Whitin algorithm is applied for each item separately. Each ant generates a sequence of items, based on the setup costs. Afterwards, a simple single stage lot-sizing rule is applied with modified setup costs. This modification of the setup costs is dependent on the item position in the lot-sizing sequence, on the already lot-sized items, and on two further parameters (tried for improvement by a systematic search). ASMLLS is among the best algorithms for small-sized problems. However, for most of the problems, which are medium and large-sized, ASMLLS performs better than all other approaches with regard to the quality of solution and the time required for computation. In 2010 Christian Almeder used A Hybrid Optimization Approach for Multi-Level Capacitated Lot-Sizing Problems.

vi) Scatter Search

In 2006, Yi Han, IkouKaku, Jiafu Tang, Nico Dellaert, Jianhu Cai, Yanlai Li and Gengui Zhou, proposed a scatter search approach for uncapacitated multilevel lot-sizing problems. One of the key problems in production planning in MRP systems is the multilevel lot-sizing

(MLLS) problem. Scatter Search (SS) has been adopted for the solution of uncapacitated MLLS problems as it provides a wide exploration of the search space through intensification and diversification. When compared with GA and AS for large size problems, this approach is not very optimistic. However, it remains as good as GA and AS and makes large improvement where compared with Wagner-Whitin algorithm (WW) over the acceptable average runtime.

vii) Binary Particle Swarm Optimization

In 2004, Fatih Taşgetiren & Yun-Chia Liang proposed a Binary PSO Algorithm for inventory lot sizing. Wagner and Whitin Algorithm has been used to solve test problems optimally (the test problems were randomly constructed). The optimal solution so obtained was compared with the solution obtained from BPSO algorithm and a traditional genetic algorithm which were coded and utilized to solve the randomly created test problems. The results of the experiment show that almost in all cases, the binary PSO algorithm is capable of finding optimal results.

Yi Hana, Jiafu Tanga, IkoKakub, LifengMua, used a particle swarm optimization with flexible inertial weight for Solving uncapacitated multilevel lot-sizing problems in 2009. PSO algorithm is used for solving the uncapacitated MLLS problem with BOM. This algorithm has been examined by comparing the experiment results with those of a genetic algorithm (GA) for checking the feasibility and effectiveness.

In 2011, Klorklear Wajanawichakon, Rapeepan Pitakaso used a binary PSO for Solving large unconstrainted multi-level lot-sizing problem and presented a binary particle swarm optimization (PSO) algorithm for unconstrained multi-level lot-sizing problem (MLLS), which is a production planning problem in materials requirements planning (MRP) system. The problem aims to find production planning which minimizes total setup costs and inventory holding costs. A binary PSO is developed to solve the problems. In this algorithm, first randomly find a set of initial solution. Then, the particles are used to find solution according to standard procedure/ methods of PSO. After that, hybrid selection mechanism is applied to restart the algorithm. Hybrid selection is a kind of general restart mechanism in PSO but beside simply restarting randomly generated new solution, introduced good solutions which are generated from standard single level lot sizing problem such as Wagner and Whitin

algorithm, Silver and Meal heuristic to combine with the current best solution and resulting in the hybrid selection to get new population which will be used as in the standard binary PSO.

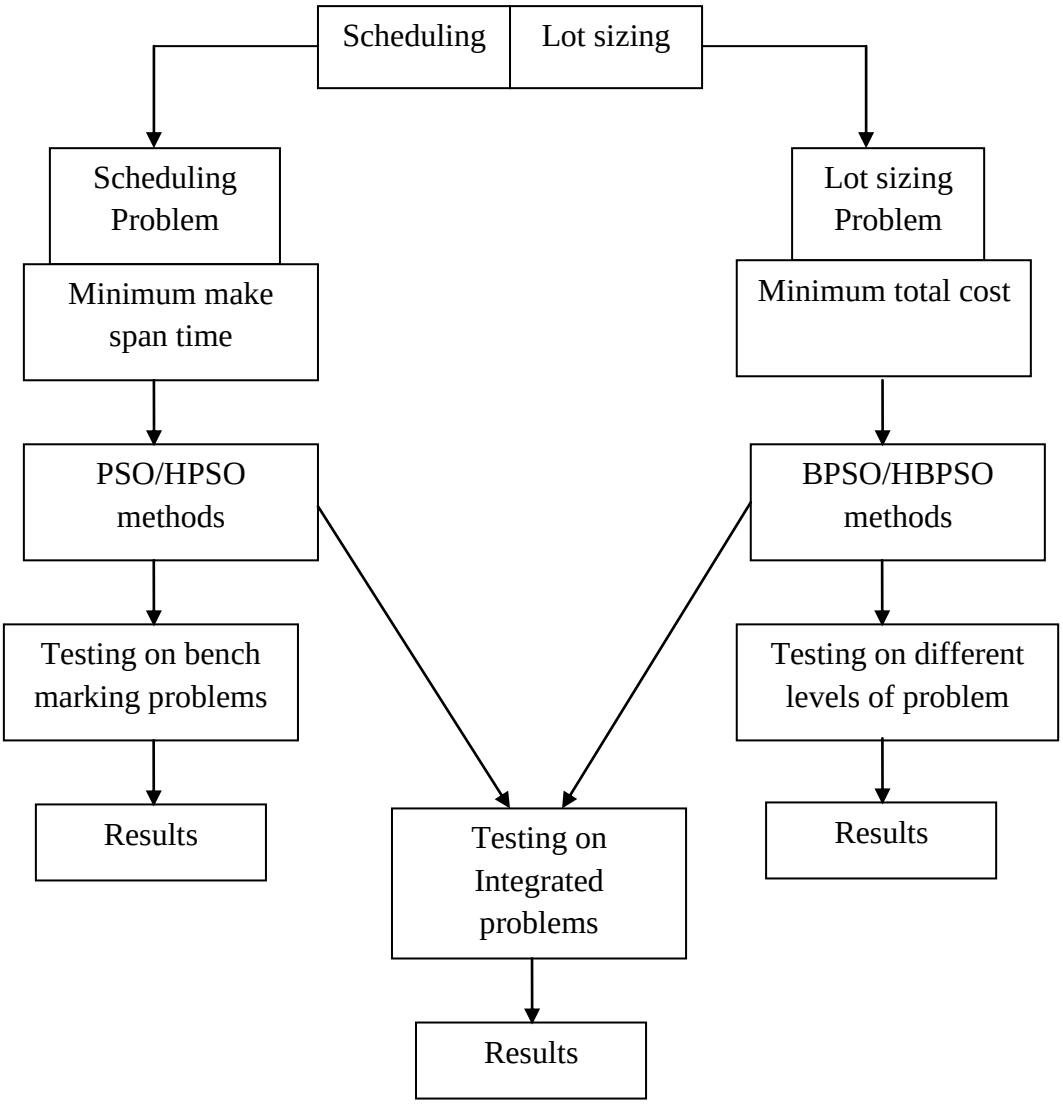
2.8. Objectives of the Work

Job shop scheduling and lot sizing problems have been considered to be gold mines for researchers as there are no unique method evolved that solve all types of scheduling and lot sizing problems. There is always a scope for developing new methods and algorithms for improvement of computational efficiency and quality of solution. Hence the present research is initiated with following objectives.

1. To implement evolutionary algorithms for Job Shop Scheduling Problems to enhance solution and computation efficiencies with respect to existing methods.
2. To implement some of the existing meta heuristics of evolutionary algorithms such as Particle Swarm Optimization (PSO), and Binary Particle Swarm Optimization algorithms to model and to solve JSSPs and lot sizing problems independently and integratedly.
3. To improvise the efficiency of these algorithms by proposing hybrids and thus to develop HPSO and HBPSO algorithms for JSSPs. The algorithms and their hybrids will be tested on all bench marking problems (LA,TA,YN, FT, ABZ, SWV, CAR DMU and ORB total to an extent of 250 Problems) to evaluate their performance visa–a–vis for scheduling.
4. To apply PSO, HPSO, BPSO and HBPSO algorithms to the different types and sizes of lot sizing problems with and without scheduling constraint. Objective is to compare total cost of the plan with and without scheduling constraints and to test for single item single level, multi level and multi item problems by applying threes algorithms and results are compared for three different sizes of the problem.
5. To develop an integrated model to solve scheduling and lot sizing problems together.
6. To develop a whole some and unique approach to solve Job Shop Scheduling and lot sizing problems of Job Shop and Batch production Industries by testing on available.

2.9. Structure of the Work Done

The work done in this thesis is schematically shown below



CHAPTER 3

SCHEDULING OF JSSPS USING PSO BASED ALGORITHMS

3.1. Introduction

The Job Shop Scheduling problem is a well-known practical planning problem in the shop floor with an objective to minimize the total completion time of all jobs i.e. from starting time of first job to completion time of last job (makespan). In most of the Job Shop Scheduling situations makespan is predominantly considered as a sole objective to schedule Jobs. The JSSP is widely acknowledged as one of the most difficult NP-complete problems. Over the last few decades, a good number of algorithms have been developed to solve JSSPs, however, no single algorithm is able to solve all kinds of JSSPs with reasonable solution in reasonable CPU time. Therefore, there is a scope for analyzing the difficulties of JSSPs as well as for the application of improved algorithms that may be able to solve them effectively.

In the present work, an attempt is made to fine tune the existing methods such as PSO and BPSO and develop new hybridized methods such as HPSO and HBPSO to solve JSSPs with different initial populations. The algorithms developed are PSO, HPSO, BPSO and HBPSO and these algorithms have been tested on wide range of Bench Marking Problems. The coding of these algorithms is done in MATLAB, optimized by speed, and run on Intel Core2Duo T6400 @ 2.00GHz and each algorithm was made to run 30 times on each problem of 250 Bench Mark Problem Instances. In this chapter, the results obtained by executing the above algorithm on 250 Bench Marking Problems are reported.

3.2. Benchmark Problems

Literature review indicates the development of variety of evolutionary algorithms and their applications to JSSPs. However, to find the comparative merits of the various techniques and algorithms, one need to test on the bench mark problems. Hence the birth of “Benchmark Problems” which provide a standard platform on which all JSSP algorithms can be tested and compared. Hence, it will be easy to gauge the strength and power of various algorithms from a statement such as “algorithm A achieved a makespan of x in time y on benchmark

problem B". As the benchmark problems are of different dimensions and grades of difficulty, it is simple to determine the capabilities and limitations of a given method by testing it on the benchmark problems. Also, the test findings may suggest the improvements required and where they should be made. These benchmark problems are formulated by various authors, **Fisher and Thompson**, 1963 **(FT)** 3 problems of 3 different sizes: 6×6, 10×10, 20×5; **Lawrence**, 1984 **(LA)** 40 problems of 8 different sizes: 10×5, 15×5, 20×5, 10×10, 15×10, 20×10, 30×10 and 15×15; **Adams Balas & Zawak**, 1988 **(ABZ)** 5 problems of 2 different sizes: 10×10, 20×15; **Applegate and Cook**, 1991 **(ORB)** 10 problems of 10×10 size; **Storer, Vaccari & Wu**, 1992 **(SWV)** 20 problems of 3 different sizes: 20×10 , 20×15, 50×10; **Yamada and Nakano**, 1992 **(YN)** 4 problems of 20×20 size; **Taillard**, 1993 **(TA)** 80 problems of 8 different sizes: 15×15, 20×15, 20×20, 30×15, 30×20, 50×15, 50×20; **Demirkol, Mehta & Uzsoy**, 1998 **(DMU)** 80 problems of 8 different sizes: 20×15, 20×20, 30×15, 30×20, 50×15, 50×20, 100×20; **Jacques Carlier**, **(CAR)** 8 problems of 8 different sizes: 11×5, 13×4, 12×5, 14×4, 10×6, 8×9, 7×7, 8×8; and are freely available. In this chapter, all the known 250 benchmark problems of JSSP have been taken into consideration for testing with the algorithms being developed.

Since all these problems are available in the website http://people.brunel.ac.uk/~mastjjb/jeb/orlib/jobshopinfo.html, http://optimizizer.com/jobshop.php, they were not included. Thus, in the present work the JSSPs have been suitably mapped in terms of PSO, HPSO, BPSO and HBPSO and with all the proposed hybridization, they were successfully implemented. The following sub section would present the methodology.

3.3. String Representation of JSSP Solution

The solution to JSSP is a schedule of operation for jobs. In the present work PSO, HPSO, BPSO and HBPSO are used to find the optimum schedule. A ***particle***, represents feasible schedule. These are similar to a chromosome representing feasible schedule in case of genetic algorithm. The solution representations show considerable variety in them. We have distinct representations for Job Shop Scheduling Problems. Those representations are *Operation-based representation* (Been, Gen, Tsujimura and Kubata et al, 1994), *Job-based representation* (Holsapple, Jacob, Pakath and Zaveru, 1993), *Preference-list-based representation* (Davis,1985; Falkenaur and Bouffouix, 1991; Croce, Tadedi and Volta, 1995; Kobayashi et al., 1995), *Job-pair-relation-based representation* (Dorndorf and

Pesch, 1995), *Disjunctive-graph-based representation* (Tamaki and Nishikawa, 1992), *Completion-time-based representation* (Nakamo and Yamada, 1992b), *Machine-based representation* (Dorndorf and Pesch, 1995) and *Random-key-based representation* (Bean 1994; Bean and Norman, 1995). The solution representations give considerable variety in them. These may be classified into the following two basic solution encoding approaches:

i. Direct approach
ii. Indirect approach

In the *direct approach*, a schedule is encoded directly into the *Particle* and the proposed algorithms are used to evolve those to discover a better schedule. The job-based representation, operation based representation, job-pair-relation- based representation, completion-time-based representation, and random key-based representation belong to this class.

In the *indirect approach*, one does not work with the schedule directly. For instance, for the priority-rule-based representation, a representation of dispatching rules for job assignment is encoded into a chromosome and the algorithm is used to evolve a better sequence of dispatching rules. Subsequently, a schedule is constructed with the sequence of dispatching rules evolved. Precedence-list-based representation, disjunctive-graph-based representation and the machine- based representation belong to this class.

In the present work, a direct approach "Operation based representation" is employed. Gen, Tsujimura and Kubota (1994) devised this representation. The schedule is represented in the form of a string. For example, for a three-job-three-machine problem, the representation would be as shown in Figure 3.1. The representation encodes a schedule as a sequence of operations, and each character of the string stands for one operation. All operations of a job are represented by the same symbol, the job number and they are interpreted according to the order of their occurrence in the sequence in which they appear in the solution string. In these cases, the string is called particle, antigen, weed, ecoli bacteria, harmony in music and the algorithms PSO, HPSO, BPSO and HBPSO are used to evolve these variables to discover potential schedules.

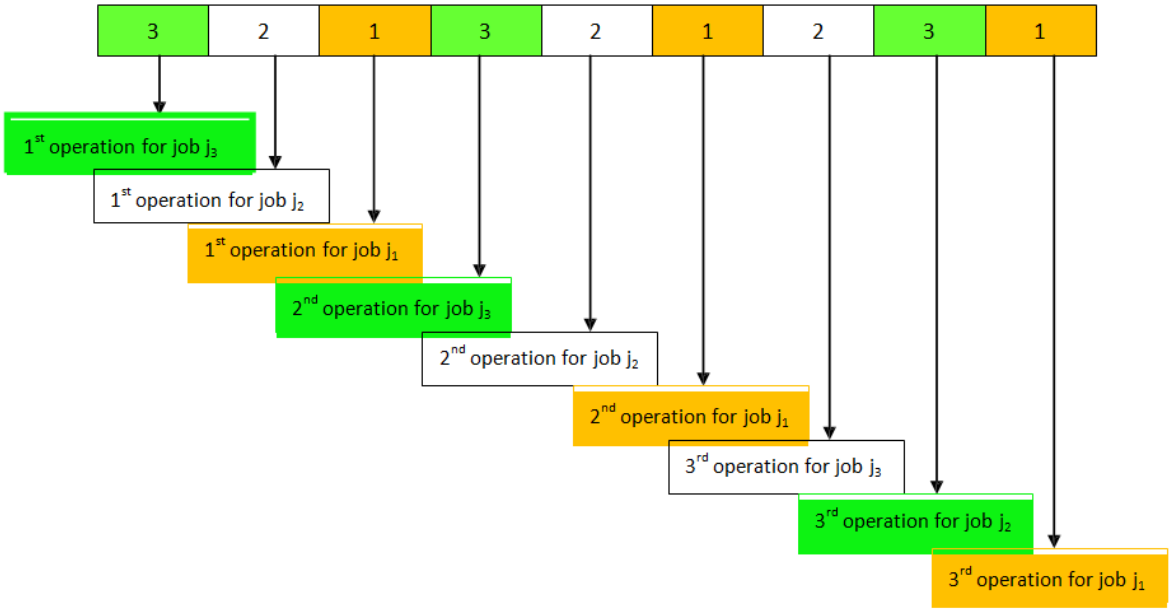


Figure 3.1 String representation of JSSP solution

3.4. Inputs to the Problem

For solving JSSP using proposed methods, the following inputs or data are required:

i. Number of jobs

ii. Number of Machines

iii. Machine order for all the jobs

iv. Processing Times of all the operations

v. Number of iterations to be carried out

vi. Maximum population allowable

For example, for a 10-*job*-5-*machine problem* the inputs should be as follows

i. Number of jobs(n) = 10

ii. Number of Machines (m) = 5,

iii. Number of operations(o) =10×5 (n×m)

All proposed methods solution efficiency depends on the generation of initial population and population size. This population size is a very important factor to find the optimal solutions

and the convergence of the solutions. In this thesis, PSO, HPSO, BPSO and HBPSO for solving Job shop objectives have been tested for all the 250 bench mark problems of different population sizes. In the preliminary experiment, we have considered 5 population sizes i.e., 20, 40, 60, 80 and 100. Along with these variations, we tested by considering the number of operations as population size. A thorough experimentation was done 30 times, for all the 250 Bench Mark Instances with different sizes of population. If the population size is more, the chances of getting optimal solution is high but it means the execution time will be more. If the population size is minimum, it means chances of getting local optimal solution is more. Therefore, in order to avoid these two situations, we fixed the population size as equal to number of operations that is to be carried out in that particular instances.

	Machine 1	Machine 2	Machine 3	Machine 4	Machine 5
Job 1	1	5	4	2	3
Job 2	4	5	2	1	3
Job 3	3	2	5	4	1
Job 4	1	4	5	2	3
Job 5	1	2	5	4	3
Job 6	3	2	5	4	1
Job 7	2	4	3	1	5
Job 8	1	2	4	5	3
Job 9	5	4	3	2	1
Job 10	2	1	3	5	4

Table 3.1. Machine order for all the jobs

	Machine 1	Machine 2	Machine 3	Machine 4	Machine 5
Job 1	13	16	19	7	14
Job 2	19	7	13	17	19
Job 3	19	18	16	18	19
Job 4	14	15	10	13	17
Job 5	8	8	19	7	9
Job 6	16	15	20	18	10
Job 7	14	17	18	5	20
Job 8	8	6	9	20	7
Job 9	16	13	9	16	12
Job 10	12	19	9	6	7

Table 3.2. Processing Time for all operations

iv. Number of iterations to be carried out (this parameter is specified at run-time depending on the complexity).

v. Population generation using selectiveness and randomness according to the problem size and the theory is already explained in Chapter 1.

Some assumptions made and used in this work are

- No one job is pre-empted
- The precedence given by the technical sequence is respected
- No two jobs are processed at the same time on the same machine
- Each job is processed to its completion, though there may be waits and delays between the operations performed.
- Jobs may be started at any time; hence no release time exists.
- Jobs must wait for the next machine to be available.
- No machine may perform more than one operation at a time.
- Set-up times for the operations are sequence-independent and included in processing times.
- There is only one of each type of machine
- Machines may be idle within the schedulable period.
- Machines are available at any time.

3.5. Particle Swarm optimization (PSO) Algorithm

One of the best evolutionary techniques for unconstrained continuous optimization is particle swarm optimization (PSO), which was proposed by Kennedy and Eberhart (1995). The solution was inspired by social behavior of bird flocking or fish schooling. PSO has been successfully used in different fields due to its ease of implementation and computational efficiency. Particles move toward the pbest position and gbest position with each iteration. The pbest position is the best position found by each particle so far. The gbest position is the best position found by the swarm so far. The particle moves itself according to its velocity. For each particle k and dimension j, the velocity and position of particles can be updated by the following equations:

For each particle k and dimension j, the velocity and position of particles can be updated by the following equations:

$$v_{kj} = w \times v_{kj} + c_1 \times rand_1 \times (pbest_{kj} - x_{kj}) + c_2 \times rand_2 \times (gbest_j - x_{kj}) \qquad (3.1)$$

$$x_{kj} = x_{kj} + v_{kj} \qquad (3.2)$$

In Equations 3.1 and 3.2, v_{kj} is the velocity of the particle k on dimension j, and x_{kj} is the position of particle k on dimension j. The $pbest_{kj}$ is the pbest position of particle k on dimension j, and $gbest_j$ is the gbest position of the swarm on dimension j. The inertia weight w is used to control exploration and exploitation. The particles maintain high velocities with a larger w, and low velocities with a smaller w. The constants c_1 and c_2 are used to decide whether particles prefer moving toward a pbest position or gbest position. The $rand_1$ and $rand_2$ are random variables between 0 and 1. The process of working of PSO through pseudo code is shown in Figure 3.2 and its flow chart is shown in Figure.3.3.

3.5.1. Encoding scheme and Initialization of swarm

One of the key issues in applying PSO successfully to JSSP is how to encode a schedule to a search solution, i.e. finding a suitable mapping of problem solution and PSO particle. Solution depends on the particle representation. Suitable particle representation should importantly impact the result and performance of PSO. A finite number of particles are being dispread over the D-dimensional problem space with random positions (initializing a population). These particles will have the ability to exchange information with its neighbor, memorize a previous position and information to take decision. The encoding scheme proposed by Zhixiong Liu, 2009 has been used in this work.

3.5.2. Fixing the Parameters

In equation 3.1, inertia weight (*w*) is an important deciding parameter for the search ability of a PSO algorithm. A large inertia weight facilitates searching a new area, whereas low inertia weight facilitates a fine searching in the current search area. Suitable selection of the inertia weight provides a balance between global exploration and local exploitation and it reduces the number of iterations to find an adequate solution. By linearly changing the inertia weight from large value to small value throughout the run, the PSO tends

to have more global search ability at the beginning of the run, while having more local search ability near the end of the run. In this thesis, the inertia weight is set to the following equation 3.3. for all computational experiments.

$$W = W_{\max} - \frac{W_{\max} - W_{\min}}{iter_{\max}} \times iter \qquad (3.3)$$

Where W_{max} and W_{min} are the initial and final values of the weighing coefficient, $iter_{max}$ is the maximum number of iteration or generations and *iter* is the current iteration or generation number. In the following computational examples, the inertia weight is ranging between 1.2 – 0.4 according to equation 3.3 throughout the run. The acceleration constants c1 and c2 are used to adjust the amount of velocities in PSO system as per eq. 3.1. Low values allow particles to roam far from target regions before being tugged back, while high values result in abrupt movement towards, or past, target regions. So from the literature and experience of other researchers [D.Y.Sha, Cheng-yu Hsu (2006), Weijun Xia ,Zhiming Wu (2006)], the acceleration constants are equal to 2.0 for all the examples. The velocity and position of the swarm is raging from [-n, n] and [1, n] respectively, where n is the number of jobs in the problem.

Begin
Step 1: Initialization
Initialize swarm, including swarm size, each particle's position and velocity;
Evaluate the each particle fitness;
Initialize gbestposition with particle with the lowest fitness in the swarm;
Initialize pbest position with a copy of particle itself;
Give initial value: $W_{max}, W_{min}, C_1, C_2$ *and generation=0;*
Step 2: Computation
While (the maximum of generation is
not met) Do {
Generation++;
Generate next swarm by equation (1a) and (1b); Evaluate Swarm
{
Find new gbest and pbest;
Update gbestof the swarm and pbest of each particle;
}

}

Step 3: Output optimization results

End

Figure 3.2. Pseudo code for Particle Swarm Optimization

3.5.3. Evaluate the fitness and update the values

After generating the population, evaluate the fitness using objective function. This fitness is used as the performance evaluation of particles in the swarm. Certain particle representation should be employed, which can establish the mapping between the scheduling solution and the particle position, and the scheduling solution can be indirectly obtained through decoding of the particle representation. In this thesis, the particle representation is based on the Operation Particle Position Sequence (OPPS).

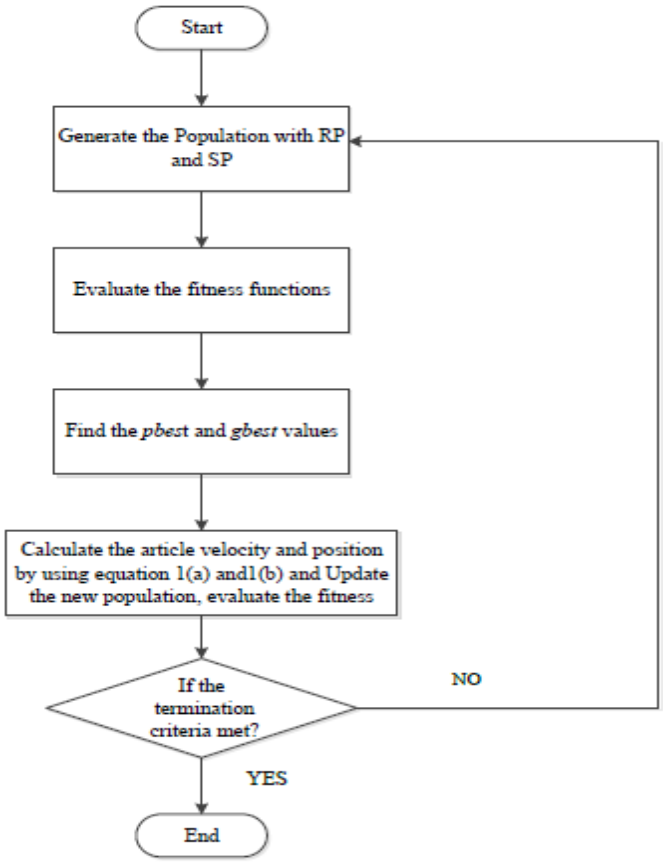


Figure 3.3. Flowchart for Particle Swarm Optimization

The feasible solutions of JSSP are the operation sequence of all jobs. For the particle position $x_i=(x_{i1}, x_{i2}, x_{i3}, \ldots\ldots x_{ij}, \ldots\ldots x_{id})$ for all position vectors x_{ij} (the total number is equal to d) also have a sequence. So the operation sequence of all jobs and the sequence of the particle position vectors can be linked together, and the mapping is gained between the scheduling solution and the particle position. JSSP consisting of m machines and n jobs, every job has to be processed on each machine, therefore, all the jobs will have (m×n) operations. Particle representation for JSSP is denoted by Table 3.3 as follows.

Operation	1	1	……	K	…..	k	…..	n
Position			……		…..		…..	

Table 3.3. Particle representation for JSSP

In Table 3.3, the length of the particle is (m×n). The first line represents all the operations of all jobs. Every operation of k-th job is numbered k, and there are m same number k in the first line. The second line denotes the i-th particle position. The sequence of all operations in first line will correspondingly change if all particle position vectors in second line are sequenced. So an operation sequence of all the jobs will be obtained.

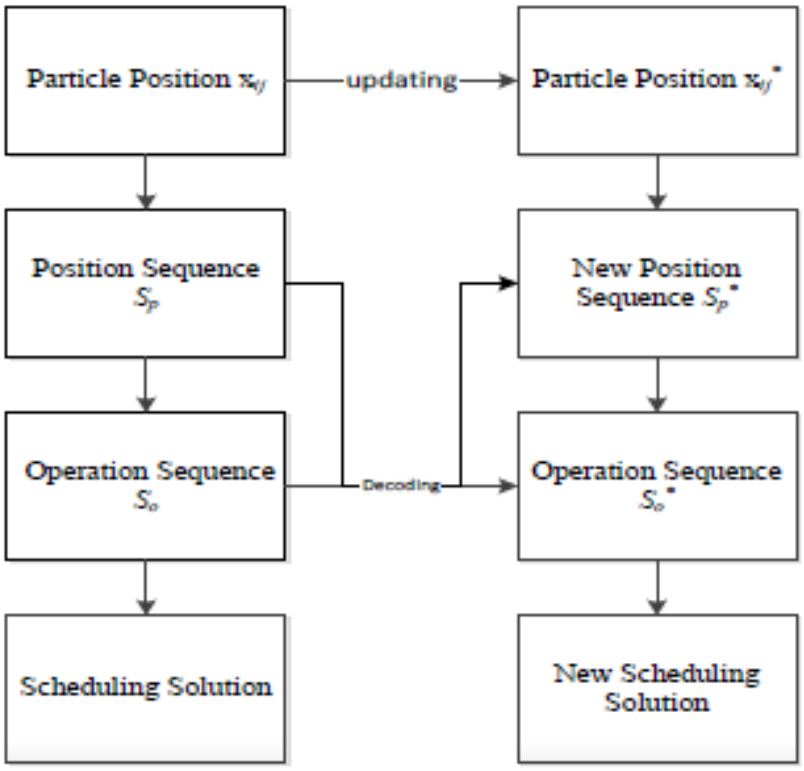


Figure 3.4. The mapping, decoding and updating of the particle representation based on OPPS

If the operation sequence of all the jobs is denoted as so and the sequence of all the particles position vector is denoted as S_p, the mapping, decoding and updating of the particle representation on Operation Particle Position Sequence (OPPS) is described by Figure 3.4. In Figure 3.4, while the particle position x_{ij} is updated to x_{ij}^*, position sequence S_p is updated to S_p^*, and then the operation sequence is changed from S_o to S_o^*. In the last step, the old scheduling solution is replaced by the new scheduling solution. During the computation of a PSO run with respect to iterations position values are replaced. In this way, updating of the particle can follow the PSO model. The global best particle and the local best particle are selected through evaluating the maximal completion time of all the jobs (makespan) for the scheduling solution. In this way the algorithm mimics the behavior of the flying birds or fishes and their means of information exchange to solve complex optimization problems.

After a thorough experimentation, the determined parameters of the PSO and HPSO; c_1 and c_2 are equal to +2 and inertia weight is w = 0.5. The inertia parameter was tested between 0 and 0.9 in increments of 0.1 and this PSO will be terminated after 10^3 iterations and HPSO will be terminated 1000 iterations. But from available literature we can run upto 10^5 iteration.

3.6. Hybrid Particle Swarm Optimization (HPSO) Algorithm

PSO is a stochastic search algorithm, it is prone to inadequate global search-ability at the end of a run. PSO may fail to find the required optima in cases when the problem to be solved is too complicated and complex. The original PSO was designed for a continuous solution space. Simulated Annealing (SA) employs certain probability to avoid becoming trapped in a local optimum and the search process can be controlled by the cooling schedule. By reasonably combining these two different search algorithms, we develop a general, fast and easily implemented hybrid optimization algorithm called HPSO. It can be seen that PSO provides initial solution for SA during the hybrid search process. Moreover, such HPSO can be applied to many combinatorial optimization problems by simple modification. In section 3.1, we have already discussed the working procedure and mapping of JSSP with

PSO. So this section briefly describes about the steps in Simulated Annealing (SA).

Begin

Step 1: Initialization

1). PSO

Initialize swarm, including swarm size, each particle's position and velocity;

Evaluate the each particle fitness;

Initialize gbest position with particle with the lowest fitness in the swarm;

Initialize pbest position with a copy of particle itself;

Give initial value: W_{max},W_{min},C_1,C_2 and generation=0;

2). SA

Determine T_o,T_{end}, B.

Step 2:Computation

1).PSO

While (the maximum of generation is not met)

Do{

Generation++;

Generate next swarm by equation (1a) and (1b);

Evaluate Swarm {

Find new gbest and pbest;

Update gbest of the swarm and pbest of each particle;

}

}

2). SA

For gbest particle S of Swarm

{$T_k = T_o$;

While ($T_k > T_{end}$)

Do {

Generate a neighbor solution S^1 from S by pair exchange method;

Compute fitness of S^1;

Evaluate S^1 {

$\Delta = f(S^1)-f(S)$;

If (min[1,exp(-Δ/T_k]>random[0,1]) {Accept S^1}

update the best solution found so far if possible;

$T_k = B*T_k$;

}

}

Step 3: Output optimization results

End

Figure 3.5. Pseudo code for Hybrid Particle Swarm Optimization (HPSO)

3.6.1. Steps in Simulated Annealing

A brief introduction on Simulated Annealing (SA) is given in Chapter 1.

Step 1: Acceptance Criteria

The resulting change "δf" in the energy of the system is computed (due to the small random displacement given to a particle) in each step of Metropolis algorithm.

If $\delta f \leq 0$, the displacement is accepted. The case $\delta f > 0$ is treated probabilistically. The probability of acceptance of the configuration is given in Equation (3.4). At each temperature, a certain number of iterations are carried out after which the temperature is decreased.

The process is repeated till the system freezes into a steady state. This equation is directly used in simulated annealing. The acceptance probability of a worse state is calculated by the equation

$$P = \exp\left\{-\frac{\delta f}{T}\right\} > r \qquad (3.4)$$

where

δf = the change in objective function

T = the current temperature

r = a random number uniformly distributed between 0 and 1.

The probability of accepting a worse move is a function of both the temperature of the system and of the change in the objective function. The probability of accepting a worse move

decreases as the temperature of the system decreases. If the temperature is zero, then only better moves will be accepted.

Step 2: Cooling Schedule

The cooling schedule of a simulated annealing algorithm consists of four components.

Starting temperature

The temperature with which the move starts should be sufficiently high so as to allow a move to almost any neighborhood state. It may be noted, unless this is done, the ending solution will be very close to the starting solution. At the same time, if the starting temperature value is too high, the search (at least in the early stages) can transform in a random search. In essence, the search will be random until the temperature is sufficiently cool to start acting as a simulated annealing algorithm. One of the methods is to quickly heat the system so that a certain worse solutions proportion are accepted and then slow cooling can start, which was suggested by Dowsland, 1995.

Final temperature

The common method is to allow the temperature to decrease till it reaches zero. This may result into the algorithm run for a lot longer. Practically, the temperature does not necessarily reach zero, since as the temperature nears zero, the probability of acceptance of a worse move are almost the same as the temperature being equal to zero. Therefore, criteria for stopping can be a suitably low temperature or otherwise when the system is frozen at the current temperature (i.e. no better or worse moves are being accepted).

Temperature decrement

Once we get the values of starting and stopping temperatures, we need to get from one to the other. To allow the system to stabilize at each temperature, enough iterations should be allowed theoretically. We need to compromise as there is an impractical situation which theoretically states that the number of iterations at each temperature to achieve this might be exponential to the problem size. This can be achieved either by performing large iterations at a few temperatures, a small number of iterations at many temperatures or a balance between the two.

Iterations at each temperature

One method is to perform a fixed number of iterations at each of the temperature. Alternatively, change the number of iterations dynamically with the progress of the algorithm. It is important to perform a large number of iterations to fully explore the local optimum at lower temperatures. The number of iterations is less at higher temperatures.

The developed PSO and HPSO algorithms were tested on 250 benchmarking instances. PSO and HPSO are working efficiently and after comparison, HPSO was observed to give better results. Table 3.4 shows the comparison of PSO, HPSO, BPSO and HBPSO with BKS values. Table 3.5 shows the % of Confirmation of the solution with BKS.

Comparison PSO, HPSO, BPSO and HBPSO with BKS by different population

S.NO	PROBLEM	BKS	**PSO**	**HPSO**	**BPSO**	**HBPSO**
1	FT06(6×6)	55	55	55	55	55
2	FT10(10×10)	930	937	930	930	930
3	FT20(20×5)	1165	1175	1176	1165	1165
4	LA01(10×5)	666	666	666	666	666
5	LA02(10×5)	655	655	655	655	655
6	LA03(10×5)	597	597	613	597	597
7	LA04(10×5)	590	590	608	590	590
8	LA05(10×5)	593	593	593	593	593
9	LA06(15×5)	926	926	943	926	926
10	LA07(15×5)	890	890	890	890	890
11	LA08(15×5)	863	863	881	863	863
12	LA09(15×5)	951	951	976	951	951
13	LA10(15×5)	958	958	969	958	958
14	LA11(20×5)	1222	1222	1222	1222	1222
15	LA12(20×5)	1039	1039	1039	1039	1039
16	LA13(20×5)	1150	1150	1150	1150	1150
17	LA14(20×5)	1292	1292	1292	1292	1292
18	LA15(20×5)	1207	1207	1207	1207	1207
19	LA16(10×10)	945	945	945	945	945
20	LA17(10×10)	784	784	784	784	784
21	LA18(10×10)	848	848	848	848	848
22	LA19(10×10)	842	842	842	842	842
23	LA20(10×10)	902	907	902	902	902
24	***LA21(15×10)***	***1046***	***1055***	***1046***	***1046***	***1046***
25	LA22(15×10)	927	935	927	927	927
26	LA23(15×10)	1032	1032	1032	1032	1032
27	***LA24(15×10)***	***935***	***937***	***935***	***935***	***935***
28	***LA25(15×10)***	***977***	***983***	***977***	***977***	***977***
29	LA26(20×10)	1218	1218	1218	1218	1218
30	***LA27(20×10)***	***1235***	***1252***	***1235***	***1235***	***1235***
31	LA28(20×10)	1216	1216	1216	1216	1216
32	***LA29(20×10)***	***1157***	***1179***	***1157***	***1163***	***1157***
33	LA30(20×10)	1355	1355	1355	1355	1355
34	LA31(30×10)	1784	1784	1784	1784	1784

35	LA32(30×10)	1850	1850	1850	1850	1850
36	LA33(30×10)	1719	1719	1719	1719	1719
37	LA34(30×10)	1721	1721	1721	1721	1721
38	LA35(30×10)	1888	1888	1888	1888	1888
39	LA36(15×15)	1268	1291	1268	1268	1268
40	LA37(15×15)	1397	1442	1397	1397	1397
41	***LA38(15×15)***	***1184***	***1228***	***1184***	***1196***	***1184***
42	LA39(15×15)	1233	1233	1233	1233	1233
43	***LA40(15×15)***	***1222***	***1236***	***1222***	***1224***	***1222***
44	ORB1(10×10)	1059	1059	1059	1059	1059
45	ORB2(10×10)	888	888	888	888	888
46	ORB3(10×10)	1005	1005	1005	1005	1005
47	ORB4(10×10)	1005	1005	1005	1005	1005
48	ORB5(10×10)	887	887	887	887	887
49	ORB6(10×10)	1010	1010	1010	1010	1010
50	ORB7(10×10)	397	397	397	397	397
51	ORB8(10×10)	899	899	899	899	899
52	ORB9(10×10)	934	934	934	934	934
53	ORB10(10×10)	944	944	944	944	944
54	SWV1(20×10)	1219	1219	1219	1219	1219
55	SWV2(20×10)	1259	1259	1259	1259	1259
56	SWV3(20×10)	1178	1178	1178	1178	1178
57	SWV4(20×10)	1161	1161	1161	1161	1161
58	SWV5(20×10)	1235	1235	1235	1235	1235
59	SWV6(20×15)	1229	1229	1229	1229	1229
60	SWV7(20×15)	1128	1128	1128	1128	1128
61	SWV8(20×15)	1330	1330	1330	1330	1330
62	SWV9(20×15)	1266	1266	1266	1266	1266
63	SWV10(20×15)	1159	1159	1159	1159	1159
64	SWV11(50×10)	2808	2808	2808	2808	2808
65	SWV12(50×10)	2829	2829	2829	2829	2829
66	SWV13(50×10)	2977	2977	2977	2977	2977
67	SWV14(50×10)	2842	2842	2842	2842	2842
68	SWV15(50×10)	2762	2762	2762	2762	2762
69	SWV16(50×10)	2924	2924	2924	2924	2924
70	SWV17(50×10)	2794	2794	2794	2794	2794
71	SWV18(50×10)	2852	2852	2852	2852	2852
72	SWV19(50×10)	2843	2843	2843	2843	2843
73	SWV20(50×10)	2823	2823	2823	2823	2823
74	ABZ5(10×10)	1234	1234	1234	1234	1234
75	ABZ6(10×10)	943	943	943	943	943
76	***ABZ7(20×15)***	***656***	***666***	***656***	***656***	***656***
77	***ABZ8(20×15)***	***645***	***655***	***645***	***645***	***645***
78	***ABZ9(20×15)***	***661***	***671***	***661***	***661***	***661***
79	YN1(20×20)	694	704	694	694	694
80	YN2(20×20)	713	723	713	713	713
81	YN3(20×20)	680	692	680	680	680
82	YN4(20×20)	719	725	719	719	719
83	TA01(15×15)	1231	1231	1231	1231	1231
84	TA02(15×15)	1244	1244	1244	1244	1244
85	TA03(15×15)	1206	1206	1206	1206	1206
86	TA04(15×15)	1170	1170	1170	1170	1170
87	TA05(15×15)	1215	1215	1215	1215	1215
88	TA06(15×15)	1210	1210	1210	1210	1210
89	TA07(15×15)	1223	1223	1223	1223	1223
90	TA08(15×15)	1187	1187	1187	1187	1187

91	TA09(15×15)	1297	1297	1297	1297	1297
92	TA10(15×15)	1241	1241	1241	1241	1241
93	TA11(20×15)	1357	1357	1357	1357	1357
94	TA12(20×15)	1367	1367	1367	1367	1367
95	TA13(20×15)	1369	1369	1369	1369	1369
96	TA14(20×15)	1345	1345	1345	1345	1345
97	TA15(20×15)	1348	1348	1348	1348	1348
98	TA16(20×15)	1351	1351	1351	1351	1351
99	TA17(20×15)	1458	1458	1458	1458	1458
100	TA18(20×15)	1412	1412	1412	1412	1412
101	TA19(20×15)	1336	1336	1336	1336	1336
102	TA20(20×15)	1347	1347	1347	1347	1347
103	TA21(20×20)	1649	1649	1649	1649	1649
104	TA22(20×20)	1627	1627	1627	1627	1627
105	TA23(20×20)	1556	1556	1556	1556	1556
106	TA24(20×20)	1624	1624	1624	1624	1624
107	TA25(20×20)	1580	1580	1580	1580	1580
108	TA26(20×20)	1672	1672	1672	1672	1672
109	TA27(20×20)	1688	1688	1688	1688	1688
110	TA28(20×20)	1602	1602	1602	1602	1602
111	TA29(20×20)	1583	1583	1583	1583	1583
112	TA30(20×20)	1573	1573	1573	1573	1573
113	TA31(30×15)	1764	1764	1764	1764	1764
114	TA32(30×15)	1774	1774	1774	1774	1774
115	TA33(30×15)	1729	1729	1729	1729	1729
116	TA34(30×15)	1828	1828	1828	1828	1828
117	TA35(30×15)	2007	2007	2007	2007	2007
118	TA36(30×15)	1819	1819	1819	1819	1819
119	TA37(30×15)	1771	1771	1771	1771	1771
120	TA38(30×15)	1673	1673	1673	1673	1673
121	TA39(30×15)	1795	1795	1795	1795	1795
122	TA40(30×15)	1631	1631	1631	1631	1631
123	TA41(30×20)	1874	1874	1874	1874	1874
124	TA42(30×20)	1867	1867	1867	1867	1867
125	TA43(30×20)	1809	1809	1809	1809	1809
126	TA44(30×20)	1927	1927	1927	1927	1927
127	TA45(30×20)	1997	1997	1997	1997	1997
128	TA46(30×20)	1940	1940	1940	1940	1940
129	TA47(30×20)	1789	1789	1789	1789	1789
130	TA48(30×20)	1912	1912	1912	1912	1912
131	TA49(30×20)	1915	1915	1915	1915	1915
132	TA50(30×20)	1807	1807	1807	1807	1807
133	TA51(50×15)	2760	2760	2760	2760	2760
134	TA52(50×15)	2756	2756	2756	2756	2756
135	TA53(50×15)	2717	2717	2717	2717	2717
136	TA54(50×15)	2839	2839	2839	2839	2839
137	TA55(50×15)	2679	2679	2679	2679	2679
138	TA56(50×15)	2781	2781	2781	2781	2781
139	TA57(50×15)	2943	2943	2943	2943	2943
140	TA58(50×15)	2885	2885	2885	2885	2885
141	TA59(50×15)	2655	2655	2655	2655	2655
142	TA60(50×15)	2723	2723	2723	2723	2723
143	TA61(50×20)	2868	2868	2868	2868	2868
144	TA62(50×20)	2869	2869	2869	2869	2869
145	TA63(50×20)	2755	2755	2755	2755	2755
146	TA64(50×20)	2702	2702	2702	2702	2702

147	TA65(50×20)	2725	2725	2725	2725	2725
148	TA66(50×20)	2845	2845	2845	2845	2845
149	TA67(50×20)	2825	2825	2825	2825	2825
150	TA68(50×20)	2784	2784	2784	2784	2784
151	TA69(50×20)	3071	3071	3071	3071	3071
152	TA70(50×20)	2995	2995	2995	2995	2995
153	TA71(100×20)	5464	5464	5464	5464	5464
154	TA72(100×20)	5181	5181	5181	5181	5181
155	TA73(100×20)	5568	5568	5568	5568	5568
156	TA74(100×20)	5339	5339	5339	5339	5339
157	TA75(100×20)	5392	5392	5392	5392	5392
158	TA76(100×20)	5342	5342	5342	5342	5342
159	TA77(100×20)	5436	5436	5436	5436	5436
160	TA78(100×20)	5394	5394	5394	5394	5394
161	TA79(100×20)	5358	5358	5358	5358	5358
162	TA80(100×20)	5183	5183	5183	5183	5183
163	DMU1(20×15)	2501	2547	2501	2501	2501
164	DMU2(20×15)	2651	2698	2651	2651	2651
165	DMU3(20×15)	2731	2776	2731	2731	2731
166	DMU4(20×15)	2601	2639	2601	2601	2601
167	DMU5(20×15)	2749	2771	2749	2749	2749
168	DMU6(20×15)	2998	3005	2998	2998	2998
169	DMU7(20×15)	2815	2854	2815	2815	2815
170	DMU8(20×15)	3051	3067	3051	3051	3051
171	DMU9(20×15)	2956	2998	2956	2956	2956
172	DMU10(20×15)	2858	2879	2858	2858	2858
173	DMU11(20×20)	3395	3415	3395	3395	3395
174	DMU12(20×20)	3418	3432	3418	3418	3418
175	DMU13(20×20)	3681	3699	3681	3681	3681
176	DMU14(20×20)	3394	3403	3394	3394	3394
177	DMU15(20×20)	3343	3365	3343	3343	3343
178	DMU16(20×20)	3734	3776	3734	3734	3734
179	DMU17(20×20)	3709	3789	3709	3709	3709
180	DMU18(20×20)	3844	3886	3844	3844	3844
181	DMU19(20×20)	3669	3684	3669	3669	3669
182	DMU20(20×20)	3604	3689	3604	3604	3604
183	DMU21(30×15)	4380	4391	4380	4380	4380
184	DMU22(30×15)	4725	4756	4725	4725	4725
185	DMU23(30×15)	4668	4689	4668	4668	4668
186	DMU24(30×15)	4648	4691	4648	4648	4648
187	DMU25(30×15)	4164	4176	4164	4164	4164
188	DMU26(30×15)	4647	4669	4647	4647	4647
189	DMU27(30×15)	4848	4881	4848	4848	4848
190	DMU28(30×15)	4692	4698	4692	4692	4692
191	DMU29(30×15)	4691	4699	4691	4691	4691
192	DMU30(30×15)	4732	4741	4732	4732	4732
193	DMU31(30×20)	5640	5651	5640	5640	5640
194	DMU32(30×20)	5927	5934	5927	5927	5927
195	DMU33(30×20)	5728	5741	5728	5728	5728
196	DMU34(30×20)	5385	5392	5385	5385	5385
197	DMU35(30×20)	5635	5642	5635	5635	5635
198	DMU36(30×20)	5621	5648	5621	5621	5621
199	DMU37(30×20)	5851	5876	5851	5851	5851
200	DMU38(30×20)	5713	5749	5713	5713	5713
201	DMU39(30×20)	5747	5769	5747	5747	5747
202	DMU40(30×20)	5577	5583	5577	5577	5577

203	DMU41(40×15)	3007	3045	3007	3007	3007
204	DMU42(40×15)	3172	3187	3172	3172	3172
205	DMU43(40×15)	3292	3301	3292	3292	3292
206	DMU44(40×15)	3283	3299	3283	3283	3283
207	DMU45(40×15)	3001	3056	3001	3001	3001
208	DMU46(40×15)	3575	3586	3575	3575	3575
209	DMU47(40×15)	3522	3564	3522	3522	3522
210	DMU48(40×15)	3447	3459	3447	3447	3447
211	DMU49(40×15)	3403	3432	3403	3403	3403
212	DMU50(40×15)	3496	3501	3496	3496	3496
213	DMU51(40×20)	3917	3943	3917	3917	3917
214	DMU52(40×20)	4065	4098	4065	4065	4065
215	DMU53(40×20)	4141	4153	4141	4141	4141
216	DMU54(40×20)	4202	4219	4202	4202	4202
217	DMU55(40×20)	4140	4163	4140	4140	4140
218	DMU56(40×20)	4554	4567	4554	4554	4554
219	DMU57(40×20)	4302	4319	4302	4302	4302
220	DMU58(40×20)	4319	4345	4319	4319	4319
221	DMU59(40×20)	4217	4242	4217	4217	4217
222	DMU60(40×20)	4319	4327	4319	4319	4319
223	DMU61(50×15)	4917	4989	4917	4917	4917
224	DMU62(50×15)	5033	5041	5033	5033	5033
225	DMU63(50×15)	5111	5122	5111	5111	5111
226	DMU64(50×15)	5130	5138	5130	5130	5130
227	DMU65(50×15)	5105	5116	5105	5105	5105
228	DMU66(50×15)	5391	5399	5391	5391	5391
229	DMU67(50×15)	5589	5589	5589	5593	5589
230	DMU68(50×15)	5426	5456	5426	5426	5426
231	DMU69(50×15)	5423	5443	5423	5423	5423
232	DMU70(50×15)	5501	5508	5501	5501	5501
233	DMU71(50×20)	6080	6081	6080	6080	6080
234	DMU72(50×20)	6395	6406	6395	6395	6395
235	DMU73(50×20)	6001	6018	6001	6001	6001
236	DMU74(50×20)	6123	6190	6123	6123	6123
237	DMU75(50×20)	6029	6201	6029	6029	6029
238	DMU76(50×20)	6342	6442	6342	6342	6342
239	DMU77(50×20)	6499	6599	6499	6499	6499
240	DMU78(50×20)	6586	6598	6586	6586	6586
241	DMU79(50×20)	6650	6670	6650	6650	6650
242	DMU80(50×20)	6459	6559	6459	6459	6459
243	CAR1(11×5)	7738	7749	7738	7738	7738
244	CAR2(13×4)	7166	7174	7166	7166	7166
245	CAR3(12×5)	7312	7342	7312	7312	7312
246	CAR4(14×4)	8003	8048	8003	8003	8003
247	CAR5(10×6)	7702	7712	7702	7702	7702
248	CAR6(8×9)	8313	8343	8313	8313	8313
249	CAR7(7×7)	6558	6588	6558	6558	6558
250	CAR8(8×8)	8264	8464	8264	8264	8264

Table 3.4 Comparison of PSO, HPSO, BPSO and HBPSO results with BKS using different population

Total problems tested = 250 Bench Mark Instances			
Method	**No. of solutions equal to BKS**	**No. of solutions not equal to BKS**	**% of Confirmation**
PSO	143	137	57.2
HPSO	246	4	98.4
BPSO	212	38	84.8
HBPSO	237	13	94.8

Table 3.5. Percentage of Confirmation of PSO, HPSO, BPSO and HBPSO solution with BKS.

3.7. Results and Analysis

In this chapter, the effectiveness and potential of the proposed and developed techniques for optimization of Job Shop Scheduling and testing of these techniques on 250 benchmarking instances are presented, analyzed and the results are compared. The developed evolutionary algorithm programs were coded in MATLAB, optimized by speed, and run on Intel Core2Duo T6400 @ 2.00GHz and each algorithm was made to run 30 times on each problem of 250 bench mark problems/instances.

3.7.1. Performance of Algorithms on FT (03) – Problems

Tables 3.6 present results of algorithms with both RP and SP respectively on FT – Bench Marking Problems.

S.NO	PROBLEM	BKS	PSO	BPSO	HPSO	HBPSO
			MAKESPAN	MAKESPAN	MAKESPAN	MAKESPAN
Fisher and Thompson (FT) 03 Problems						
1	FT06 (6*6)	**55**	55	55	55	55
2	FT10 (10*10)	**930**	**937**	930	**930**	930
3	FT20 (20*5)	**1165**	**1175**	1165	**1176**	1165

Table 3.6: Comparison between BKS and all proposed EAs for FT Problems.

It is found that while testing on FT problems (03), BPSO and HBPSO are giving equal results with that of BKS. The solution with AIA and PSO with RP and SP are different on FT20 problem with BKS. HPSO had generated 1175 units of makespan against BKS of 1165 units.

The results in the form of number of solutions equal to BKS on FT problems are shown in Figure 3.6.

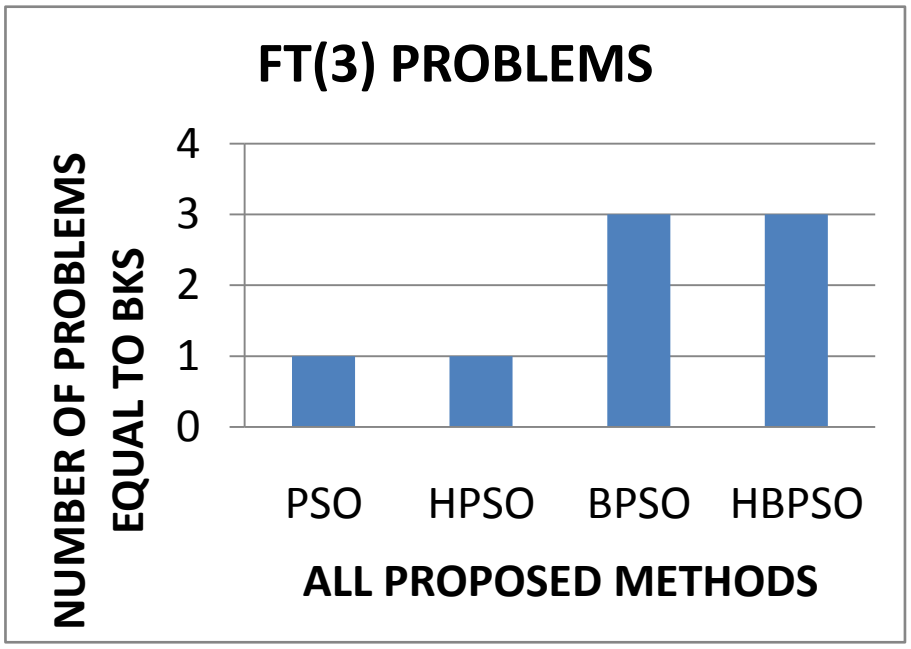


Figure.3.6 Comparison of FT, number of problems equal to BKS by all proposed methods

3.7.2. Performance of Algorithms on LA (40) – Problems

The result of PSO, HPSO, BPSO and HBPSO algorithms an LA (40) sets of problems is given in Table 3.7. The result is also compared with BKS.

S.NO	PROBLEM	BKS	PSO MAKESPAN	BPSO MAKESPAN	HPSO MAKESPAN	HBPSO MAKESPAN
Lawrence (LA) 40 Problems						
1	LA01 (10*5)	666	666	666	666	666
2	LA02 (10*5)	655	655	655	655	655
3	LA03 (10*5)	597	597	597	**613**	597
4	LA04 (10*5)	590	590	590	**608**	590
5	LA05 (10*5)	593	593	593	593	593
6	LA06 (15*5)	926	926	926	**943**	926
7	LA07 (15*5)	890	890	890	890	890
8	LA08 (15*5)	863	863	863	**881**	863
9	LA09 (15*5)	951	951	951	**976**	951
10	LA10 (15*5)	958	958	958	**969**	958

11	LA11 (20*5)	1222	1222	1222	1222	1222
12	LA12 (20*5)	1039	1039	1039	1039	1039
13	LA13 (20*5)	1150	1150	1150	1150	1150
14	LA14 (20*5)	1292	1292	1292	1292	1292
15	LA15 (20*5)	1207	1207	1207	1207	1207
16	LA16 (10*10)	945	945	945	945	945
17	LA17 (10*10)	784	784	784	784	784
18	LA18 (10*10)	848	848	848	848	848
19	LA19 (10*10)	842	842	842	842	842
20	LA20 (10*10)	902	907	902	902	902
21	LA21 (15*10)	***1046***	***1055***	***1046***	***1046***	***1046***
22	LA22 (15*10)	927	**935**	927	927	927
23	LA23 (15*10)	1032	1032	1032	1032	1032
24	LA24 (15*10)	***935***	***937***	***935***	***935***	***935***
25	LA25 (15*10)	***977***	***983***	***977***	***977***	***977***
26	LA26 (20*10)	1218	1218	1218	1218	1218
27	LA27 (20*10)	***1235***	***1252***	***1235***	***1235***	***1235***
28	LA28 (20*10)	1216	1216	1216	1216	1216
29	LA29 (20*10)	***1157***	***1179***	***1163***	***1157***	***1157***
30	LA30 (20*10)	1355	1355	1355	1355	1355
31	LA31 (20*10)	1784	1784	1784	1784	1784
32	LA32 (20*10)	1850	1850	1850	1850	1850
33	LA33 (20*10)	1719	1719	1719	1719	1719
34	LA34 (20*10)	1721	1721	1721	1721	1721
35	LA35 (20*10)	1888	1888	1888	1888	1888
36	LA36 (15*15)	1268	**1291**	1268	1268	1268
37	LA37 (15*15)	1397	**1442**	1397	1397	1397
38	LA38 (15*15)	***1184***	***1228***	***1196***	***1184***	***1184***
39	LA39 (15*15)	1233	1233	1233	1233	1233
40	LA40 (15*15)	***1222***	***1236***	***1224***	***1222***	***1222***

Table 3.7: Comparison between the BKS and all proposed EAs for LA Problems.

It is found that BPSO is giving equal results to BKS on all 40 problems i.e. it produces 100% results equal to BKS on all these LA series problems. PSO and HPSO are giving only 75% and 85% results equal to BKS respectively, whereas BPSO and HBPSO are giving 90% and 85% equal results with BKS respectively. Figure 3.7 shows the result of number of problems equal to BKS by various algorithms. Table 3.8 depicts the consolidated result of all the algorithms with number of problem solutions equal to BKS.

Method	No. of solutions equal to BKS	No. of solutions not equal to BKS	% of Confirmation
PSO	30	10	75.0
HPSO	37	03	92.5
BPSO	34	06	85.0
HBPSO	40	0	100

Table 3.8 Consolidated results of all algorithm

It is found that out of Lawrence 40 problems, PSO produced same solutions for 30 problems; BPSO produced same solution for 37 problems, similarly HPSO produced same solutions for 34 problems and HBPSO produced same solution for 40 problems. Hence, HBPSO has been found to be most suitable to solve Lawrence-40 problems or similar problems.

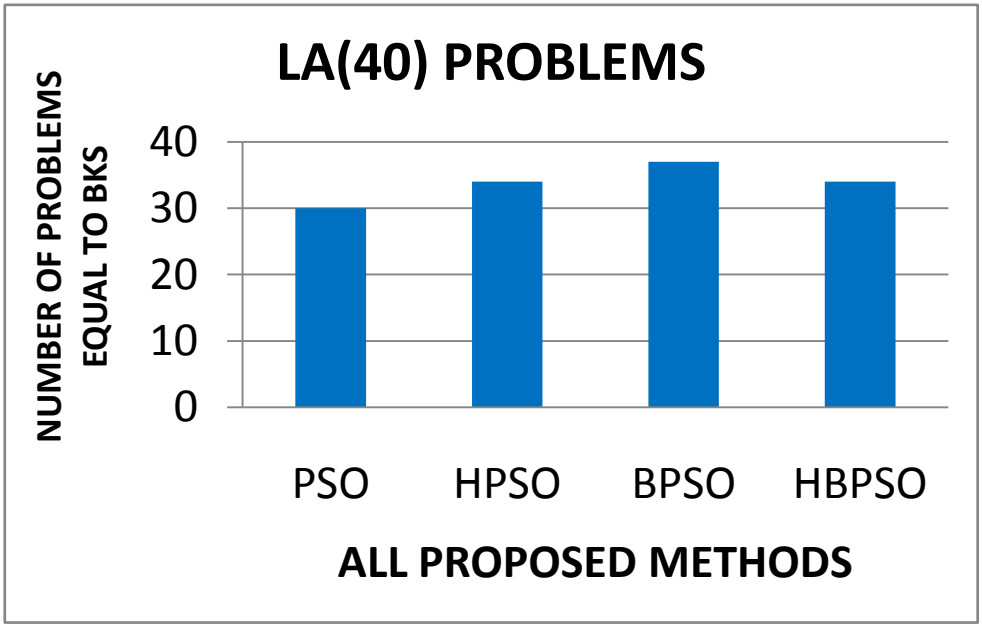


Figure.3.7. Comparison of LA, number of problems equal to BKS by all proposed methods

3.7.3. Performance of Algorithms on ORB (10) - Problems

Bench Mark Instances of ORB (10) problems were also solved by the developed algorithms and performance of these algorithms are presented in Table: 3.9.

S.NO	PROBLEM	BKS	PSO MAKESPAN	BPSO MAKESPAN	HPSO MAKESPAN	HBPSO MAKESPAN
Applegate and Cook (ORB) 10 Problems						
1	ORB1 (10*10)	**1059**	1059	1059	1059	1059
2	ORB2 (10*10)	**888**	888	888	888	888
3	ORB3 (10*10)	**1005**	1005	1005	1005	1005
4	ORB4 (10*10)	**1005**	1005	1005	1005	1005
5	ORB5 (10*10)	**887**	887	887	887	887
6	ORB6 (10*10)	**1010**	1010	1010	1010	1010
7	ORB7 (10*10)	**397**	397	397	397	397
8	ORB8 (10*10)	**899**	899	899	899	899
9	ORB9 (10*10)	**934**	934	934	934	934
10	ORB10 (10*10)	**944**	944	944	944	944

Table 3.9 Comparison between the BKS and all proposed EAs for ORB Problems

It is observed that PSO based algorithms performance on ORB (10) problems are excellent as the solution generated by these algorithms are 100% equal to BKS.

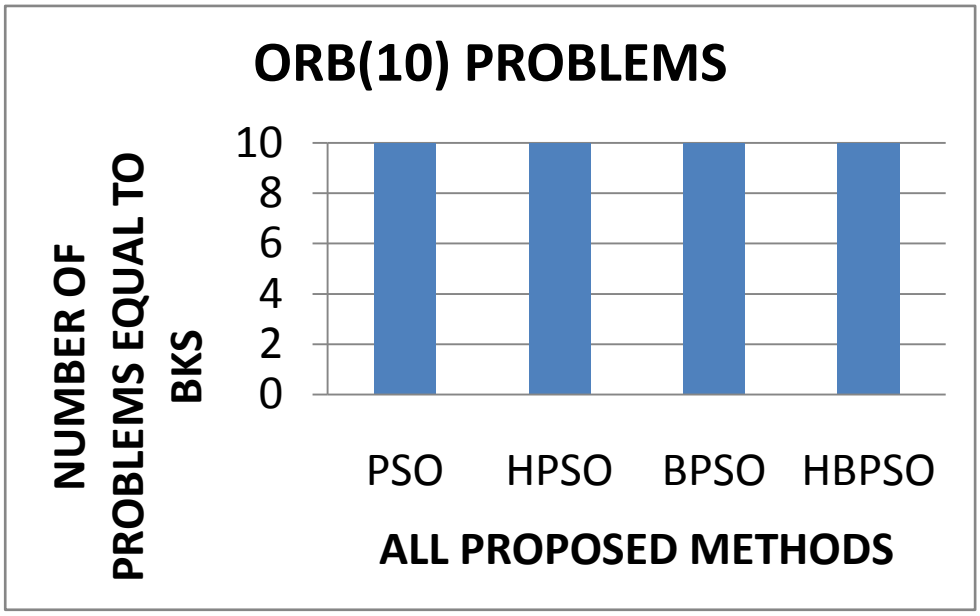


Figure.3.8. Comparison of ORB, number of problems equal to BKS by all proposed methods

3.7.4. Performance of Algorithms on SWV (20) – Problems

The make span of each problem of SWV – Bench Mark Instances performed by each algorithm are presented in Table 3.10.

S.NO	PROBLEM	BKS	PSO MAKESPAN	BPSO MAKESPAN	HPSO MAKESPAN	HBPSO MAKESPAN
Storer, Vaccari & Wu (SWV) 20 Problems						
1	SWV1 (20*10)	**1219**	1219	1219	1219	1219
2	SWV2 (20*10)	**1259**	1259	1259	1259	1259
3	SWV3 (20*10)	**1178**	1178	1178	1178	1178
4	SWV4 (20*10)	**1161**	1161	1161	1161	1161
5	SWV5 (20*10)	**1235**	1235	1235	1235	1235
6	SWV6 (20*15)	**1229**	1229	1229	1229	1229
7	SWV7 (20*15)	**1128**	1128	1128	1128	1128
8	SWV8 (20*15)	**1330**	1330	1330	1330	1330
9	SWV9 (20*15)	**1266**	1266	1266	1266	1266

10	SWV10 (20*15)	**1159**	1159	1159	1159	1159
11	SWV11 (50*10)	**2808**	2808	2808	2808	2808
12	SWV12 (50*10)	**2829**	2829	2829	2829	2829
13	SWV13 (50*10)	**2977**	2977	2977	2977	2977
14	SWV14 (50*10)	**2842**	2842	2842	2842	2842
15	SWV15 (50*10)	**2762**	2762	2762	2762	2762
16	SWV16 (50*10)	**2924**	2924	2924	2924	2924
17	SWV17 (50*10)	**2794**	2794	2794	2794	2794
18	SWV18 (50*10)	**2852**	2852	2852	2852	2852
19	SWV19 (50*10)	**2843**	2843	2843	2843	2843
20	SWV20 (50*10)	**2823**	2823	2823	2823	2823

Table 3.10 Comparison between the BKS and all proposed EAs for SWV Problems.

Further, it is very interesting to find that PSO based algorithms developed in this work are generating solutions (i.e. make span) equal to BKS. Hence, PSO based algorithm, PSO, HPSO, BPSO and HBPSO performance is 100% on SWV (20) problems.

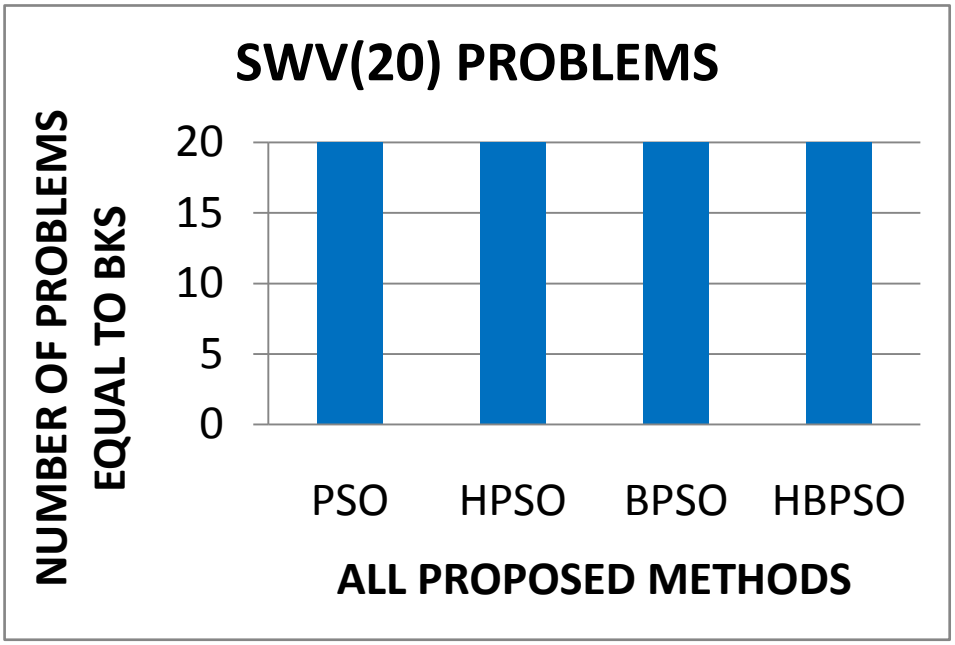


Figure.3.9. Comparison of SWV, number of problems equal to BKS by all proposed methods

3.7.5. Performance of Algorithms on ABZ (05) – Problems

Table 3.11 depicts the result of algorithms on 5 ABZ problems.

S.NO	PROBLEM	BKS	PSO MAKESPAN	BPSO MAKESPAN	HPSO MAKESPAN	HBPSO MAKESPAN
Adams Balas & Zawak (ABZ) 05 Problems						
1	ABZ5 (10*10)	**1234**	1234	1234	1234	1234
2	ABZ6 (10*10)	**943**	943	943	943	943
3	ABZ7 (20*15)	***656***	***666***	***656***	***656***	***656***
4	ABZ8 (20*15)	***645***	***655***	***645***	***645***	***645***
5	ABZ9 (20*15)	***661***	***671***	***661***	***661***	***661***

Table 3.11 Comparison between the BKS and all proposed EAs for ABZ Problems using RP.

From Table 3.11, it is found that while testing on ABZ problems (10), PSO, HPSO, BPSO and HBPSO are giving equal results with BKS, whereas, PSO performance is equal to BKS of two problems only. Thus, 40% results are equal to BKS. The results were shown in Figure 3.10.

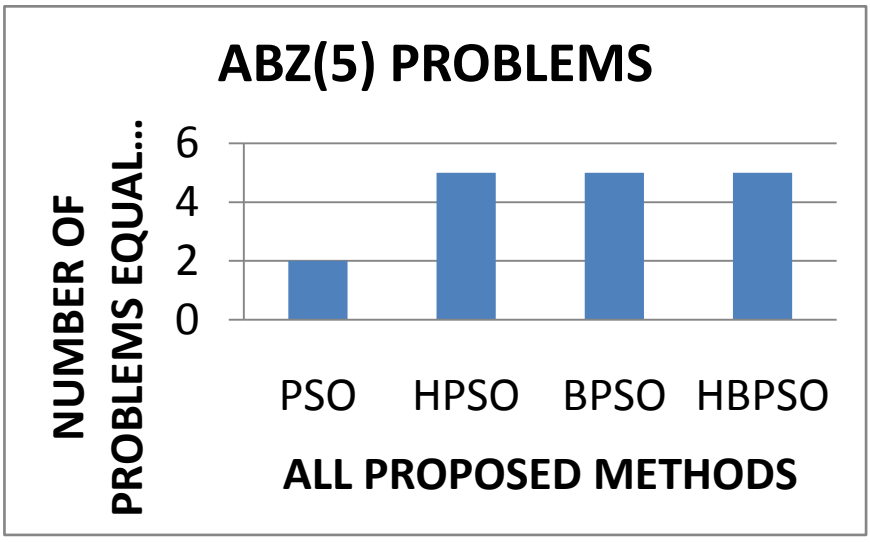


Figure.3.10 Comparison of ABZ, number of problems equal to BKS by all proposed methods

3.7.6. Performance of Algorithms on YN (04) – Problems

Table 3.12 depicts the result of PSO based algorithms on 4 YN problems.

S.NO	PROBLEM	BKS	PSO MAKESPAN	BPSO MAKESPAN	HPSO MAKESPAN	HBPSO MAKESPAN
Yamada and Nakano (YN) 04 Problems						
1	YN1 (20*20)	**694**	**704**	694	694	694
2	YN2 (20*20)	**713**	**723**	713	713	713
3	YN3 (20*20)	**680**	**692**	680	680	680
4	YN4 (20*20)	**719**	**725**	719	719	719

Table 3.12 Comparison between the BKS and all proposed EAs for YN Problems.

From Table 3.12, it is found that PSO is not able to produce BKS for YN (4) problems. Whereas, HPSO, BPSO and HBPSO have generated 100% solution equal to BKS. The result is shown as bar graph in Figure 3.11.

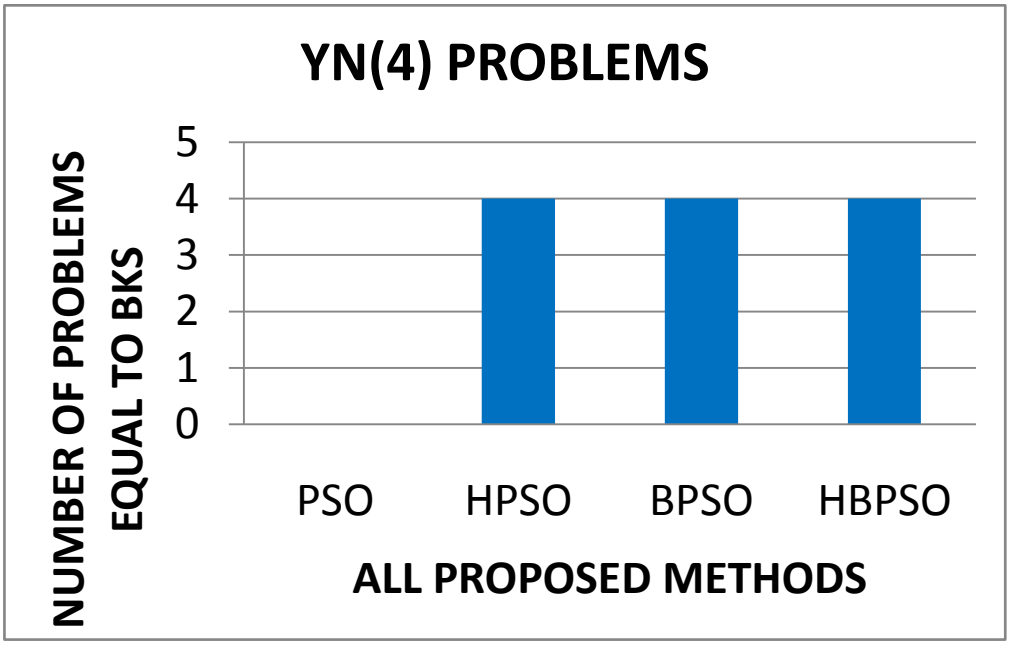


Figure 3.11 Comparison of YN, number of problems equal to BKS by all proposed methods

3.7.7. Performance of Algorithms on TA (80) – Problems

Tillard had cited 80 problems as bench mark instances TA (80). The performance of the algorithm developed is tested on TA(80) problems and result is shown in Table 3.13.

S.NO	PROBLEM	BKS	PSO MAKESPAN	HPSO MAKESPAN	HPSO MAKESPAN	HBPSO MAKESPAN
Taillard (TA) 80 Problems						
1	TA01 (15*15)	**1231**	1231	1231	1231	1231
2	TA02 (15*15)	**1244**	1244	1244	1244	1244
3	TA03 (15*15)	**1206**	1206	1206	1206	1206
4	TA04 (15*15)	**1170**	1170	1170	1170	1170
5	TA05 (15*15)	**1215**	1215	1215	1215	1215
6	TA06 (15*15)	**1210**	1210	1210	1210	1210
7	TA07 (15*15)	**1223**	1223	1223	1223	1223
8	TA08 (15*15)	**1187**	1187	1187	1187	1187
9	TA09 (15*15)	**1297**	1297	1297	1297	1297

10	TA10 (15*15)	**1241**	1241	1241	1241	1241
11	TA11 (20*15)	**1357**	1357	1357	1357	1357
12	TA12 (20*15)	**1367**	1367	1367	1367	1367
13	TA13 (20*15)	**1369**	1369	1369	1369	1369
14	TA14 (20*15)	**1345**	1345	1345	1345	1345
15	TA15 (20*15)	**1348**	1348	1348	1348	1348
16	TA16 (20*15)	**1351**	1351	1351	1351	1351
17	TA17 (20*15)	**1458**	1458	1458	1458	1458
18	TA18 (20*15)	**1412**	1412	1412	1412	1412
19	TA19 (20*15)	**1336**	1336	1336	1336	1336
20	TA20 (20*15)	**1347**	1347	1347	1347	1347
21	TA21 (20*20)	**1649**	1649	1649	1649	1649
22	TA22 (20*20)	**1627**	1627	1627	1627	1627
23	TA23 (20*20)	**1556**	1556	1556	1556	1556
24	TA24 (20*20)	**1624**	1624	1624	1624	1624
25	TA25 (20*20)	**1580**	1580	1580	1580	1580
26	TA26 (20*20)	**1672**	1672	1672	1672	1672
27	TA27 (20*20)	**1688**	1688	1688	1688	1688
28	TA28 (20*20)	**1602**	1602	1602	1602	1602
29	TA29 (20*20)	**1583**	1583	1583	1583	1583
30	TA30 (20*20)	**1573**	1573	1573	1573	1573
31	TA31 (30×15)	**1764**	1764	1764	1764	1764
32	TA32 (30×15)	**1774**	1774	1774	1774	1774
33	TA33 (30×15)	**1729**	1729	1729	1729	1729
34	TA34 (30×15)	**1828**	1828	1828	1828	1828
35	TA35 (30×15)	**2007**	2007	2007	2007	2007
36	TA36 (30×15)	**1819**	1819	1819	1819	1819
37	TA37 (30×15)	**1771**	1771	1771	1771	1771
38	TA38 (30×15)	**1673**	1673	1673	1673	1673

39	TA39 (30×15)	**1795**	1795	1795	1795	1795
40	TA40 (30×15)	**1631**	1631	1631	1631	1631
41	TA41 (30×20)	**1874**	1874	1874	1874	1874
42	TA42 (30×20)	**1867**	1867	1867	1867	1867
43	TA43 (30×20)	**1809**	1809	1809	1809	1809
44	TA44 (30×20)	**1927**	1927	1927	1927	1927
45	TA45 (30×20)	**1997**	1997	1997	1997	1997
46	TA46 (30×20)	**1940**	1940	1940	1940	1940
47	TA47 (30×20)	**1789**	1789	1789	1789	1789
48	TA48 (30×20)	**1912**	1912	1912	1912	1912
49	TA49 (30×20)	**1915**	1915	1915	1915	1915
50	TA50 (30×20)	**1807**	1807	1807	1807	1807
51	TA51 (50×15)	**2760**	2760	2760	2760	2760
52	TA52 (50×15)	**2756**	2756	2756	2756	2756
53	TA53 (50×15)	**2717**	2717	2717	2717	2717
54	TA54 (50×15)	**2839**	2839	2839	2839	2839
55	TA55 (50×15)	**2679**	2679	2679	2679	2679
56	TA56 (50×15)	**2781**	2781	2781	2781	2781
57	TA57 (50×15)	**2943**	2943	2943	2943	2943
58	TA58 (50×15)	**2885**	2885	2885	2885	2885
59	TA59 (50×15)	**2655**	2655	2655	2655	2655
60	TA60 (50×15)	**2723**	2723	2723	2723	2723
61	TA61 (50×20)	**2868**	2868	2868	2868	2868
62	TA62 (50×20)	**2869**	2869	2869	2869	2869
63	TA63 (50×20)	**2755**	2755	2755	2755	2755
64	TA64 (50×20)	**2702**	2702	2702	2702	2702
65	TA65 (50×20)	**2725**	2725	2725	2725	2725
66	TA66 (50×20)	**2845**	2845	2845	2845	2845
67	TA67 (50×20)	**2825**	2825	2825	2825	2825
68	TA68 (50×20)	**2784**	2784	2784	2784	2784
69	TA69 (50×20)	**3071**	3071	3071	3071	3071

70	TA70 (50×20)	**2995**	2995	2995	2995	2995
71	TA71 (100×20)	**5464**	5464	5464	5464	5464
72	TA72 (100×20)	**5181**	5181	5181	5181	5181
73	TA73 (100×20)	**5568**	5568	5568	5568	5568
74	TA74 (100×20)	**5339**	5339	5339	5339	5339
75	TA75 (100×20)	**5392**	5392	5392	5392	5392
76	TA76 (100×20)	**5342**	5342	5342	5342	5342
77	TA77 (100×20)	**5436**	5436	5436	5436	5436
78	TA78 (100×20)	**5394**	5394	5394	5394	5394
79	TA79 (100×20)	**5358**	5358	5358	5358	5358
80	TA80 (100×20)	**5183**	5183	5183	5183	5183

Table 3.13 Comparison between the BKS and all proposed EAs for TA Problems.

The following observation is made:

Method	**No. of solutions equal to BKS**	**No. of solutions not equal to BKS**	**% of Confirmation**
PSO	80	0	100
HPSO	80	0	100
BPSO	80	0	100
HBPSO	80	0	100

Hence it is concluded that PSO, BPSO, HPSO and HBPSO algorithms have excellently executed all TA (80) problems and generated schedules equal to BKS.

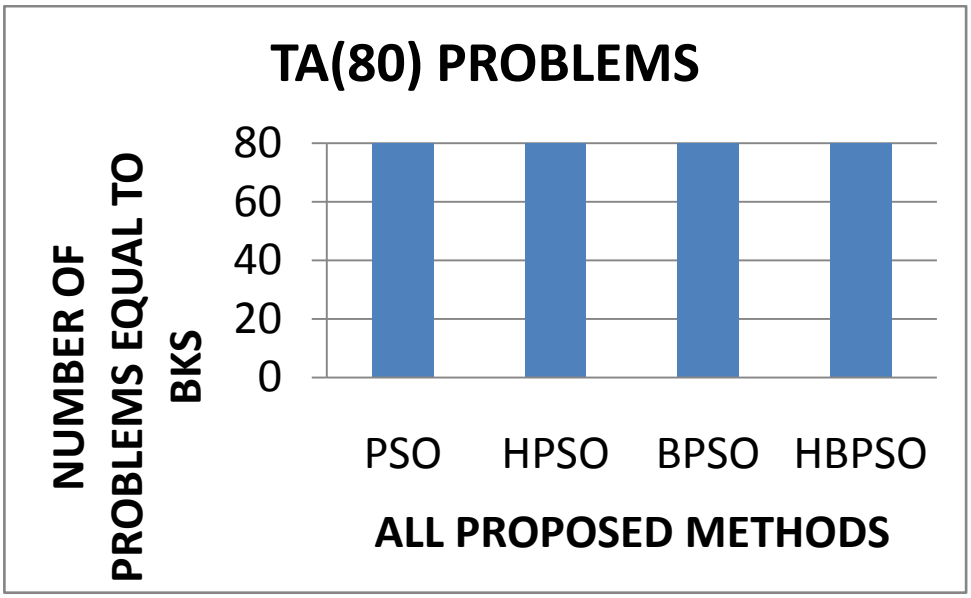


Figure.3.12. Comparison of TA, number of problems equal to BKS by all proposed methods

3.7.8. Performance of Algorithms on DMU (80) - Problems

Table: 3.14 depict the result of algorithms executed on Bench marking instances of 80 DMU problems.

S.NO	PROBLEM	BKS	PSO MAKESPAN	BPSO MAKESPAN	HPSO MAKESPAN	HBPSO MAKESPAN
Demirkol, Mehta & Uzsoy (DMU) 80 Problems						
1	DMU1 (20x15)	**2501**	**2547**	2501	2501	2501
2	DMU2 (20x15)	**2651**	**2698**	2651	2651	2651
3	DMU3 (20x15)	**2731**	**2776**	2731	2731	2731
4	DMU4 (20x15)	**2601**	**2639**	2601	2601	2601
5	DMU5 (20x15)	**2749**	**2771**	2749	2749	2749
6	DMU6 (20x15)	**2998**	**3005**	2998	2998	2998
7	DMU7 (20x15)	**2815**	**2854**	2815	2815	2815
8	DMU8 (20x15)	**3051**	**3067**	3051	3051	3051
9	DMU9 (20x15)	**2956**	**2998**	2956	2956	2956
10	DMU10 (20x15)	**2858**	**2879**	2858	2858	2858
11	DMU11 (20x20)	**3395**	**3415**	3395	3395	3395
12	DMU12 (20x20)	**3418**	**3432**	3418	3418	3418
13	DMU13 (20x20)	**3681**	**3699**	3681	3681	3681
14	DMU14 (20x20)	**3394**	**3403**	3394	3394	3394

15	DMU15 (20×20)	**3343**	**3365**	3343	3343	3343
16	DMU16 (20×20)	**3734**	**3776**	3734	3734	3734
17	DMU17 (20×20)	**3709**	**3789**	3709	3709	3709
18	DMU18 (20×20)	**3844**	**3886**	3844	3844	3844
19	DMU19 (20×20)	**3669**	**3684**	3669	3669	3669
20	DMU20 (20×20)	**3604**	**3689**	3604	3604	3604
21	DMU21 (20×15)	**4380**	**4391**	4380	4380	4380
22	DMU22 (20×15)	**4725**	**4756**	4725	4725	4725
23	DMU23 (20×15)	**4668**	**4689**	4668	4668	4668
24	DMU24 (20×15)	**4648**	**4691**	4648	4648	4648
25	DMU25 (20×15)	**4164**	**4176**	4164	4164	4164
26	DMU26 (20×15)	**4647**	**4669**	4647	4647	4647
27	DMU27 (20×15)	**4848**	**4881**	4848	4848	4848
28	DMU28 (20×15)	**4692**	**4698**	4692	4692	4692
29	DMU29 (20×15)	**4691**	**4699**	4691	4691	4691
30	DMU30 (20×15)	**4732**	**4741**	4732	4732	4732
31	DMU31 (20×20)	**5640**	**5651**	5640	5640	5640
32	DMU32 (20×20)	**5927**	**5934**	5927	5927	5927
33	DMU33 (20×20)	**5728**	**5741**	5728	5728	5728
34	DMU34 (20×20)	**5385**	**5392**	5385	5385	5385
35	DMU35 (20×20)	**5635**	**5642**	5635	5635	5635
36	DMU36 (20×20)	**5621**	**5648**	5621	5621	5621
37	DMU37 (20×20)	**5851**	**5876**	5851	5851	5851
38	DMU38 (30×20)	**5713**	**5749**	5713	5713	5713
39	DMU39 (20×20)	**5747**	**5769**	5747	5747	5747
40	DMU40 (20×20)	**5577**	**5583**	5577	5577	5577
41	DMU41 (40×15)	**3007**	**3045**	3007	3007	3007
42	DMU42 (40×15)	**3172**	**3187**	3172	3172	3172
43	DMU43 (40×15)	**3292**	**3301**	3292	3292	3292
44	DMU44 (40×15)	**3283**	**3299**	3283	3283	3283
45	DMU45 (40×15)	**3001**	**3056**	3001	3001	3001
46	DMU46 (40×15)	**3575**	**3586**	3575	3575	3575

47	DMU47 (40×15)	**3522**	**3564**	3522	3522	3522
48	DMU48 (40×15)	**3447**	**3459**	3447	3447	3447
49	DMU49 (40×15)	**3403**	**3432**	3403	3403	3403
50	DMU50 (40×15)	**3496**	**3501**	3496	3496	3496
51	DMU51 (40×20)	**3917**	**3943**	3917	3917	3917
52	DMU52 (40×20)	**4065**	**4098**	4065	4065	4065
53	DMU53 (40×20)	**4141**	**4153**	4141	4141	4141
54	DMU54 (40×20)	**4202**	**4219**	4202	4202	4202
55	DMU55 (40×20)	**4140**	**4163**	4140	4140	4140
56	DMU56 (40×20)	**4554**	**4567**	4554	4554	4554
57	DMU57 (40×20)	**4302**	**4319**	4302	4302	4302
58	DMU58 (40×20)	**4319**	**4345**	4319	4319	4319
59	DMU59 (40×20)	**4217**	**4242**	4217	4217	4217
60	DMU60 (40×20)	**4319**	**4327**	4319	4319	4319
61	DMU61 (50×15)	**4917**	**4989**	4917	4917	4917
62	DMU62 (50×15)	**5033**	**5041**	5033	5033	5033
63	DMU63 (50×15)	**5111**	**5122**	5111	5111	5111
64	DMU64 (50×15)	**5130**	**5138**	5130	5130	5130
65	DMU65 (50×15)	**5105**	**5116**	5105	5105	5105
66	DMU66 (50×15)	**5391**	**5399**	5391	5391	5391
67	DMU67 (50×15)	**5589**	**5589**	5593	5589	5589
68	DMU68 (50×15)	**5426**	**5456**	5426	5426	5426
69	DMU69 (50×15)	**5423**	**5443**	5423	5423	5423
70	DMU70 (50×15)	**5501**	**5508**	5501	5501	5501
71	DMU71 (50×20)	**6080**	**6080**	6080	6080	6080
72	DMU72 (50×20)	**6395**	**6406**	6395	6395	6395
73	DMU73 (50×20)	**6001**	**6018**	6001	6001	6001
74	DMU74 (50×20)	**6123**	**6190**	6123	6123	6123
75	DMU75 (50×20)	**6029**	**6201**	6029	6029	6029
76	DMU76 (50×20)	**6342**	**6442**	6342	6342	6342
77	DMU77 (50×20)	**6499**	**6599**	6499	6499	6499
78	DMU78 (50×20)	**6586**	**6598**	6586	6586	6586

79	DMU79 (50×20)	**6650**	**6670**	6650	6650	6650
80	DMU80 (50×20)	**6459**	**6559**	6459	6459	6459

Table 3.14 Comparison between the BKS and all proposed EAs for DMU Problems

From the above table, we found that BPSO, HPSO and HBPSO generated solutions on all DMU(80) problems that are equal to BKS (100% yield). Whereas, in the case of PSO, only two solutions are equal to BKS. Therefore, PSO has shown poor performance on DMU (80) problems.

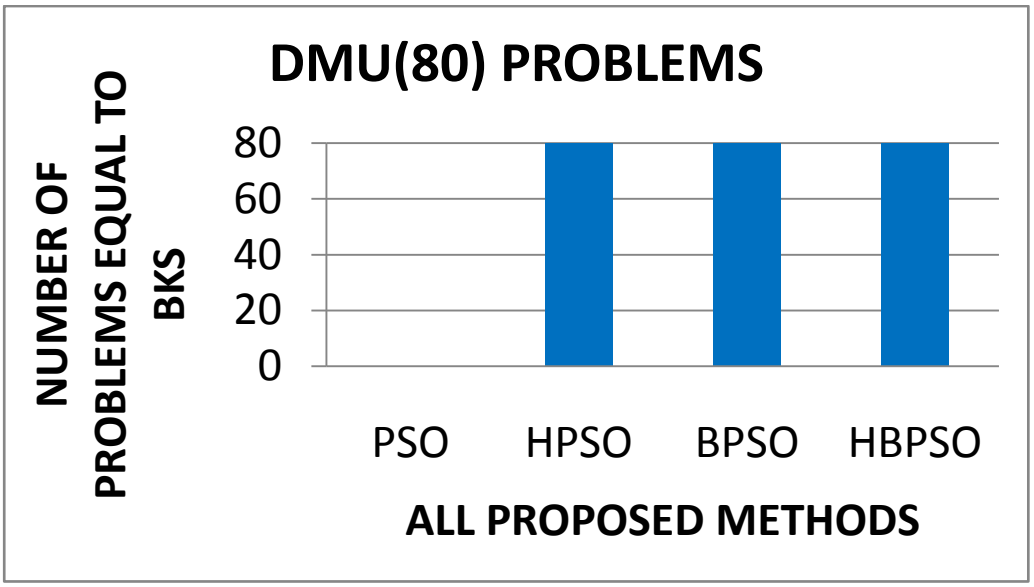


Figure.3.13 Comparison of DMU, number of problems equal to BKS by all proposed methods

3.7.9. Performance of Algorithms on CAR (08) – Problems

Table 3.15 depicts the result of algorithms on CAR (08) problems and a comparison between the BKS and all proposed algorithms. We found that the algorithms developed in this work i.e. PSO, BPSO, HPSO and HBPSO are generating same solution equal to BKS on all CAR (08) problems. The same is shown in Figure 3.14.

S.NO	PROBLEM	BKS	PSO MAKESPAN	BPSO MAKESPAN	HPSO MAKESPAN	HBPSO MAKESPAN
Jacques Carlier (CAR) 8 Problems						
1	CAR1 (11*5)	**7738**	**7749**	7738	7738	7738
2	CAR2 (13*4)	**7166**	**7174**	7166	7166	7166
3	CAR3 (12*5)	**7312**	**7342**	7312	7312	7312
4	CAR4 (14*4)	**8003**	**8048**	8003	8003	8003
5	CAR5 (10*6)	**7702**	**7712**	7702	7702	7702
6	CAR6 (8*9)	**8313**	**8343**	8313	8313	8313
7	CAR7 (7*7)	**6558**	**6588**	6558	6558	6558
8	CAR8 (8*8)	**8264**	**8464**	8264	8264	8264

Table 3.15 Comparison between the BKS and all proposed EAs for CAR Problems.

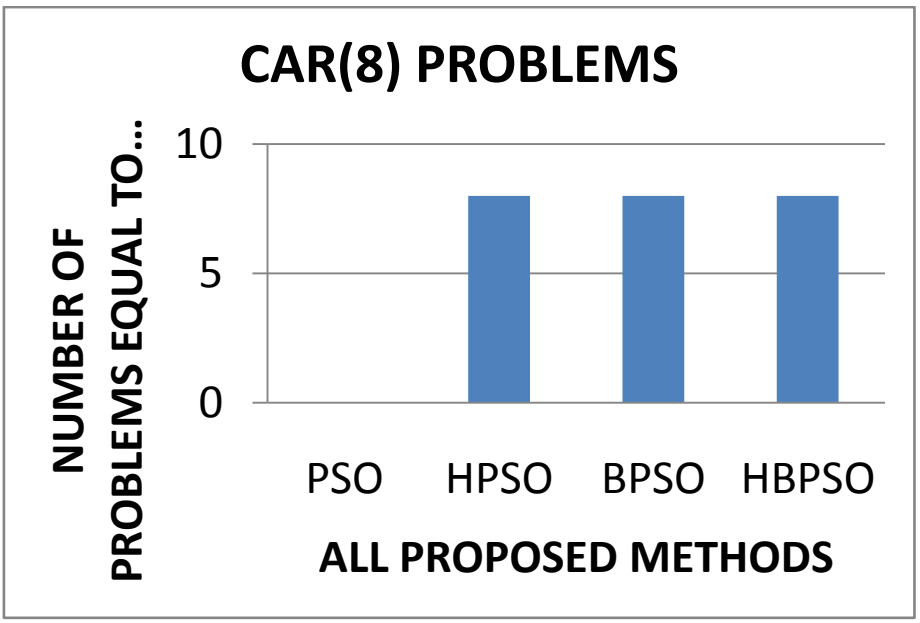


Figure.3.14 Comparison of CAR, number of problems equal to BKS by all proposed methods

TOTAL NUMBER OF BENCH MARK INSTANCES = 250																													
Problems	**FT (03)**			**LA(40)**			**ORB(10)**			**SWV(20)**			**ABZ(5)**			**YN(4)**			**TA(80)**			**DMU(80)**			**CA**				
Methods	**=**	**≠**	**%**	**=**	**≠**	**%**	**=**	**≠**	**%**	**=**	**≠**	**%**	**=**	**≠**	**%**	**=**	**≠**	**%**	**=**	**≠**	**%**	**=**	**≠**	**%**	**=**	**≠**	**%**		
PSO	1	2	33.	30	10	75%	10	0	**100**	20	0	**100**	2	3	4	0	4	0	80		**1**	0	8	0%	0	8	0		
BPSO	3	0	**100**	36	4	90%	10	0	**100**	20	0	**100**	5	0	**1**	4	0	**1**	80	0	**1**	80	0	**10**	8	0	**1**		
HPSO	1	2	33.	34	6	85%	10	0	**100**	20	0	**100**	5	0	**1**	4	0	**1**	80	0	**1**	80	0	**10**	8	0	**1**		
HBPSO	3	0	**100**	34	6	85%	10	0	**100**	20	0	**100**	5	0	**1**	4	0	**1**	80	0	**1**	80	0	**10**	8	0	**1**		

Table 3.16 Comparison of % of Improvement table for different problems

From this Table 3.16, **"="** represents the no. of problems equal to BKS, **"#"** represents the no. of problems not equal to BKS, **"% of ="** represents the % of Equivalence with the BKS.

3.7.10. Testing on Ten Tough Bench Mark Problems (Yamada)

The two well-known benchmark problems with sizes of 10×10 and 20×5 (known as FT10 and FT20) formulated by J. Fisher Muth and Thompson are commonly used as test beds to measure the effectiveness of certain methods. The FT10 problem used to be called a "notorious" problem, because it remained unsolved for over 20 years. The effectiveness of our methods on FT series problem is already shown in Table 3.16 and Figure 3.16. Applegate and Cook proposed a set of benchmark problems called the "Ten Tough Problems" as a more difficult computational challenge than the FT10 problem by collecting difficult problems from the literature (Yamada, 1997). These Ten tough problems are compared with BKS and Multi Step Cross over Fusion- GA (MSXF–GA) method developed by Yamada. Tables 3.17 show the comparison of all proposed algorithms on Ten Tough Problems and comparison with MSXF-GA method.

S. No	Problem	Size (n×m)	Best Known Solution	MSXF-GA	PSO	BPSO	HPSO	HBPSO
				Make span	Make span	Make span	Make span	Make span
1	ABZ7	20×15	**656**	692.5	**666**	656	656	656
2	ABZ8	20×15	**645**	703.1	**655**	645	645	645
3	ABZ9	20×15	**661**	719.6	**671**	661	661	661
4	LA21	15×10	**1046**	1049.9	**1055**	1046	1046	1046
5	LA24	15×10	**935**	938.8	**937**	935	935	935
6	LA25	20×10	**977**	979.6	**983**	977	977	977
7	LA27	20×10	**1235**	1253.6	**1252**	1235	1235	1235

8	LA29	20×10	**1157**	1181.9	**1179**	1157	1163	1157
9	LA38	15×15	**1184**	1198.4	**1228**	1184	1196	1184
10	LA40	15×15	**1222**	1227.9	**1236**	1222	**1224**	1222

Table 3.17 Comparison of Ten Tough Problems with BKS and All Proposed EAs.

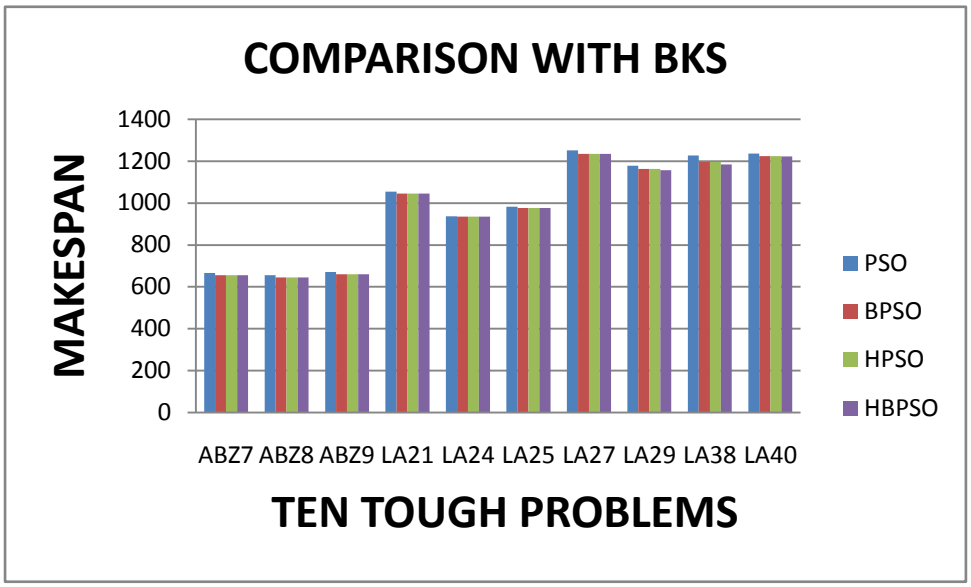


Figure 3.15 Comparison of BKS with MSXF-GA, PSO, BPSO, HPSO, HBPSO.

3.7.11. Effect of Hybridization

Among all the algorithms, two algorithms were hybridized. Original PSO was hybridized with SA (HPSO), BPSO is hybridized with (IIT) thus HBPSO was developed. It is found that hybrid algorithms are showing improved performance compared to original algorithms. Table 3.18 shows the effect of hybrid algorithms. From this study we conclude that, hybrid algorithm performance is better than the base algorithms (BPSO performance is better than PSO). By applying the hybridization, explorative power of the methods will be improved and they do not get trapped in the local optima situation for all the problems. Hence, hybrid algorithm improves and works better than the original algorithms.

No. of Bench Mark Problems Tested = 250					
No. of solutions equal to BKS by PSO & HPSO, BPSO & HBPSO					
PSO	HPSO	% Improvement	BPSO	HBPSO	% Improvement
143	246	41.86	212	237	12

Table 3.18 Effect of Hybridization

3.7.12. Convergence of Algorithms

Depending on the working principles and methodology, each algorithm will take its own time for execution. All the algorithms need to be converged for generating optimum results in a reasonable amount of time. A convergence criterion is very important to test the time efficiency of the algorithms. It is very difficult to give convergence graphs for all the 250 tested problems. Hence, convergence graphs of 10-tough problems mentioned by Yamada(1995) were given in Figures from Figure 3.16 to Figure 3.19. A careful study of these graphs would indicate number of iterations required for an algorithm to terminate (as listed in Table 3.19). PSO, HPSO, BPSO and HBPSO will be terminated in 1000 iterations.

Algorithm	**No. of Iterations**
PSO	900
BPSO	800
HPSO	800
HBPSO	700

Table 3.19 Number of Iterations required for convergence of solution

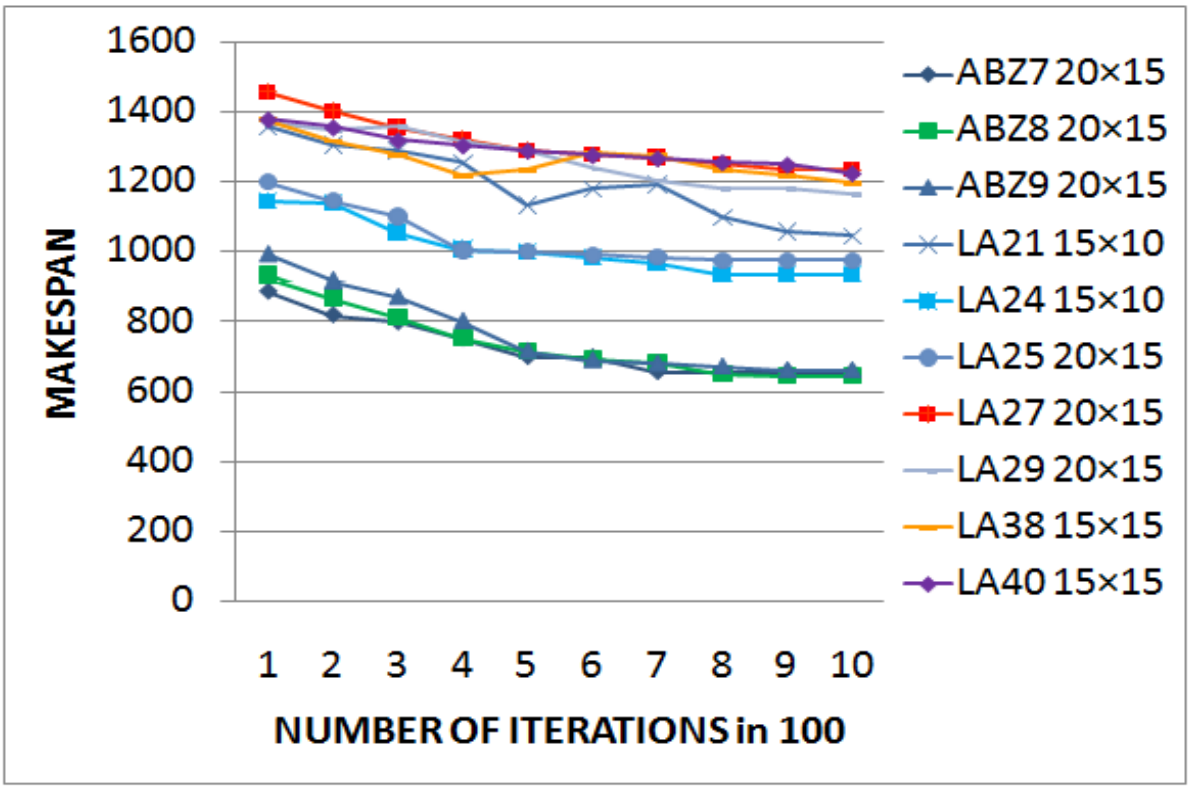


Figure 3.16 Convergence graph for PSO.

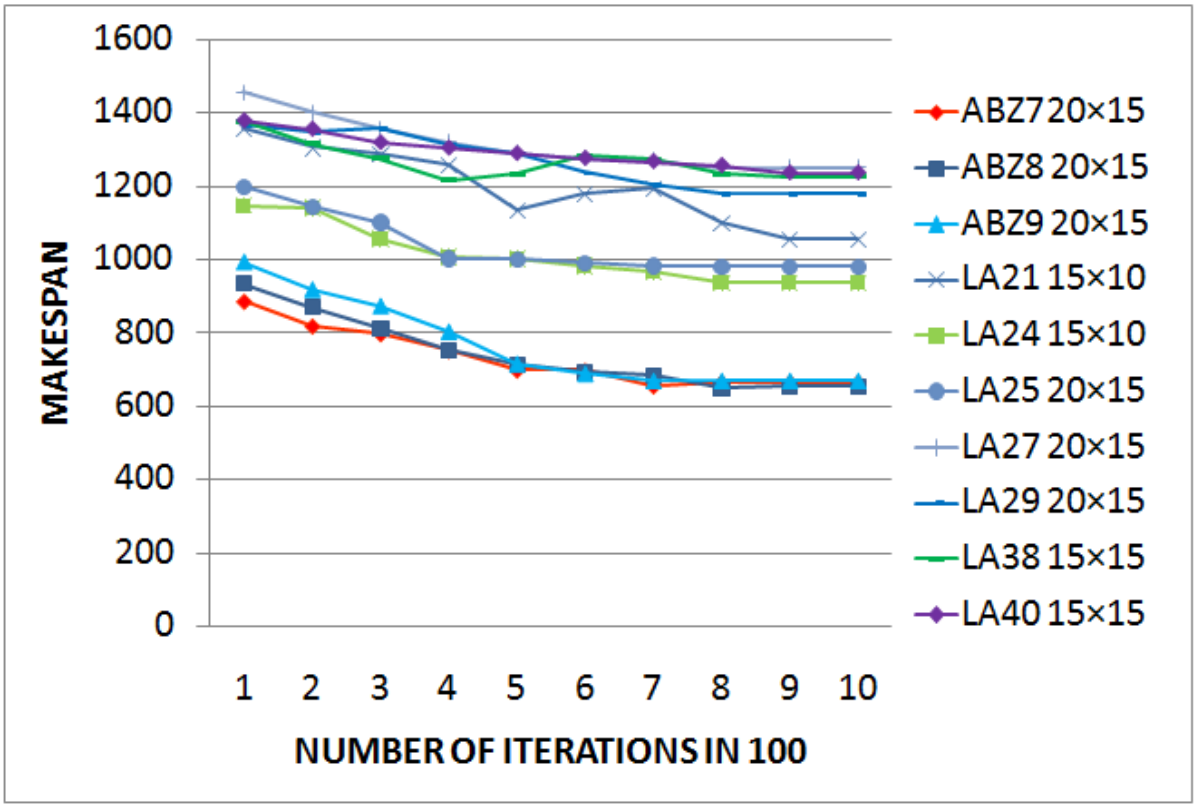


Figure 3.17 Convergence graph for HPSO.

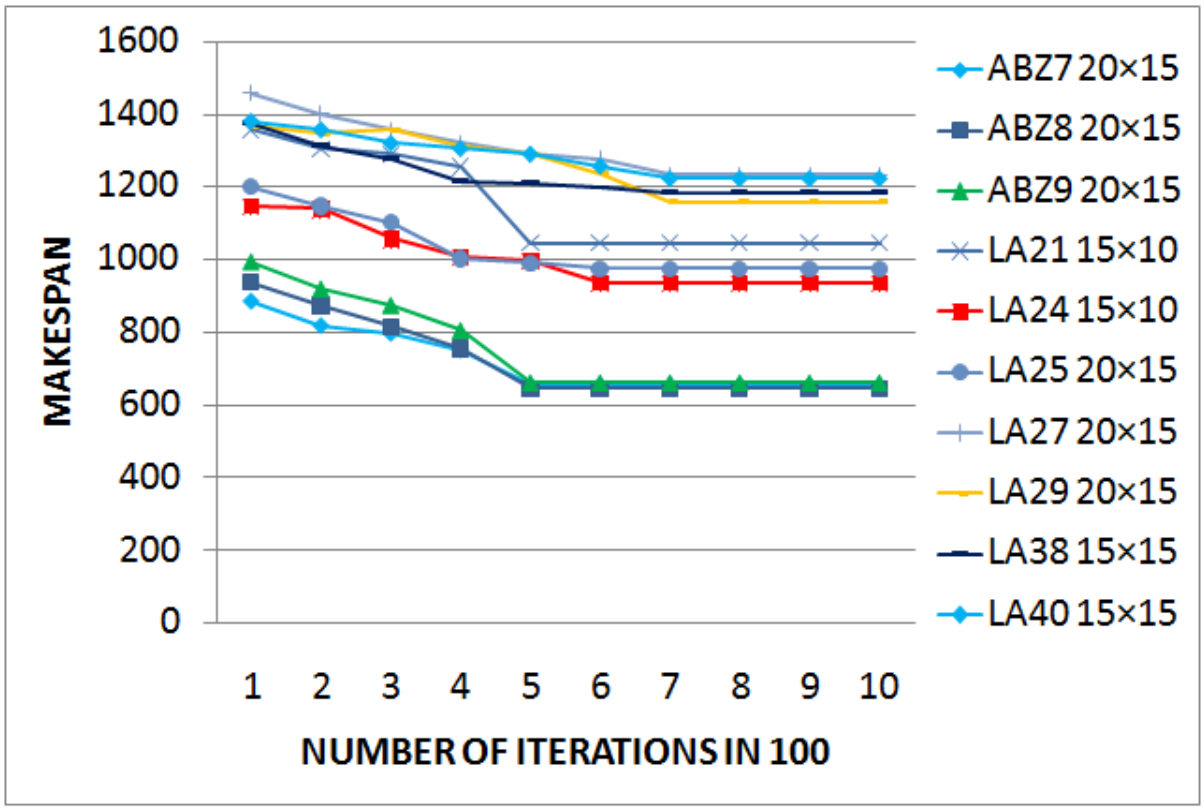


Figure 3.18 Convergence graph for BPSO

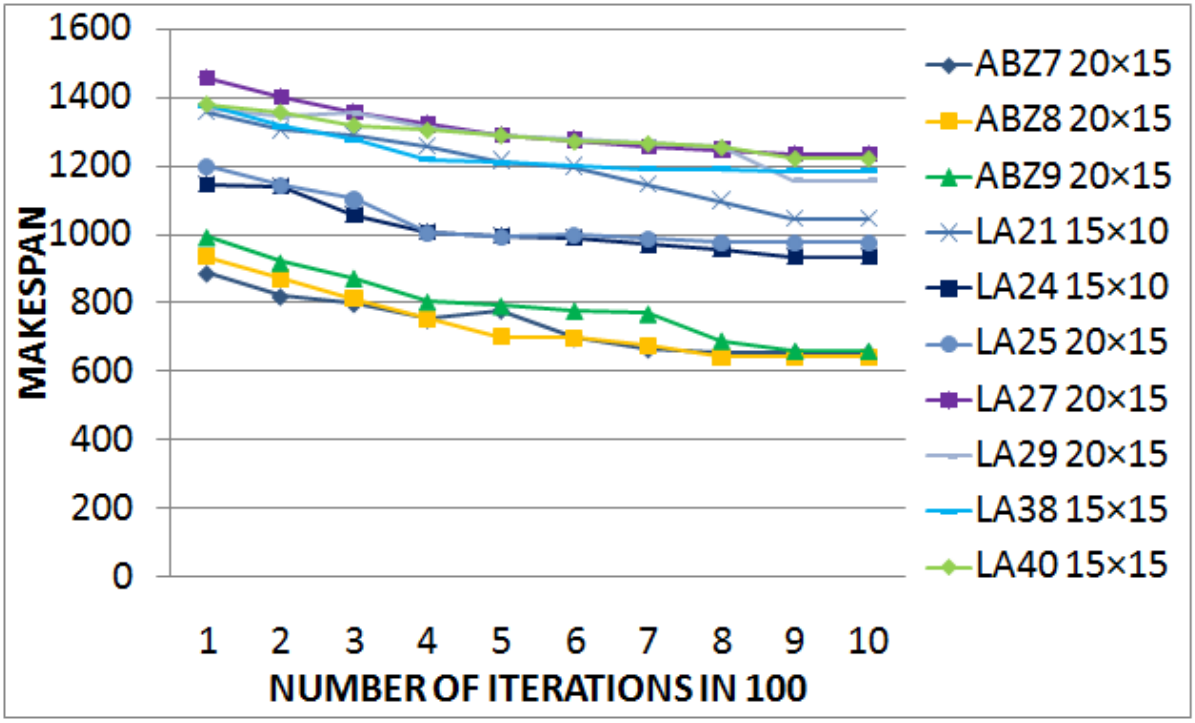


Figure 3.19 Convergence graph for HBPSO.

3.8. Summary

In this chapter, the soft computational work experience on optimization makespan time in Job shop scheduling is presented. The effectiveness and potentiality of proposed algorithms namely PSO, BPSO, HPSO and HBPSO were thoroughly tested on 250 bench mark problems including FT(3), LA(40), ORB(10), SWV(20), ABZ(5), YN(4), TA(80), DMU(80) and CAR(8). Interestingly we found that performance of HBPSO in solving all bench mark problems is excellent as it is able to generate solutions equal to Best Known Solution (BKS) for 100% bench mark problems. At the same time, HPSO, BPSO are also good tools to address JSSPs. The performance of PSO on bench mark problems is poor. Hence, this investigation concludes that HBPSO would be an efficient unique tool to solve any type of JSSP.

CHAPTER 4

PSO, BPSO, HPSO AND HBPSO APPROACH FOR LOT SIZING

4.1. Introduction

Lot sizing problem can be explained as following:

There are N number of items for production in 'T' periods. The planning is to be done in such a way so as to meet the forecasted demand. In multistage production systems, the planning of each item depends on the production of lower level items. There is a limitation with respect to production/ set up resources. Lead times are taken as zero and shortages are not allowed. Also, it is assumed that the demands are deterministic. Mathematical models for different lot sizing models take the following form:

4.2. Mathematical Formulation of SILS problem

One of the simplest model in lot sizing problem is of single item (with single level) being produced in a facility with unlimited capacity and no shortages. Mathematical representation for this type of model (LSP) is as below:

$$\min\left(\sum_{i=1}^{n} Ax_i + cI_i\right) \qquad (1)$$

subject to:

$$I_0 = 0 \qquad (2)$$

$$I_{i-1} + x_i Q_i - I_i = R_i \quad \forall i \qquad (3)$$

$$I_i \geq 0 \quad \forall i \qquad (4)$$

$$Q_i \geq 0 \quad \forall i \qquad (5)$$

$$x_i \in \{0,1\} \quad \forall i \qquad (6)$$

where:

n	number of periods
A	setup cost per order
c	carrying cost per unit per period
R_i	requirements for period i
Q_i	order quantity for period i
I_i	ending inventory for period i
x_i	is 1 if an order is placed in period i, 0 otherwise

In the objective function (i) Penalty A is levied for each order placed and penalty c per unit is levied for each item which is carried over to the next period as inventory. The opening inventory balance is zero which is specified by Equation 2, Equation (3) attempts to satisfy the net requirements and Qi (order quantity) includes the overall requirement till the next order is placed. Equation (4) is the restriction which ensures that inventory values are either zero or more than zero i.e. item shortages are not allowed. Equation (5) is the restriction which ensures that order quantities are either zero or more than zero, and Equation (6) ensures that the decision variable x_i to be 0 or 1 so that an order is to be placed if the decision variable is 1 and vice-versa if the value is 0. Given that, inventory opening balance is zero, $I_0 = 0$, it is observed that $z_1 = 1$ by Equation (3), if $R_1 > 0$. As the nature of the problem is minimization, the closing inventory at each period is minimized so as to avoid the penalty charge c, particularly I = 0.

4.3. Mathematical formulation uncapacitated MLLS problem

The mathematical model of MLLS can be formulated as follows:

$$\min \sum_{i=1}^{P} \sum_{t=1}^{T} (s_i y_{i,t} + h_i I_{i,t}) \qquad (1)$$

$$\text{s.t.} \begin{cases} I_{i,t} = I_{i,t-1} + x_{i,t} - d_{i,t} & (2) \\ d_{i,t} = \sum_{j \in \Gamma(i)} c_{i,j} x_{j,t} & (3) \\ x_{i,t} - M y_{i,t} \leq 0,\ y_{i,t} \in \{0,1\} & (4) \\ I_{i,t} \geq 0,\ x_{i,t} \geq 0 & (5) \end{cases}$$

$C_{i,j}$: quantity of item i required to produce one unit of items j.

$D_{i,j}$: external requirement for items i in period t.

H_i : holding cost for items i (Following small instance standard).

$I_{i,0}$: initial inventory of product i.

S_i : setup cost for items i (Following small instance standard).

T : total number of periods.

Decision and auxiliary variables:

$D_{i,j}$: total requirement for item i in period t.

$I_{i,t}$: Inventory level of item i at the end of period t.

$X_{i,t}$: delivered quantity of items i at the beginning of the period t.

$y_{i,t}$: binary variable which indicates if an item i is

produced in period t, ($y_{i,t} = 1$) or not ($y_{i,t}= 0$).

The aim is to minimize the aggregate of inventory holding cost and set-up cost for all items over the planning horizon in Eq. (1). The constraints on inventory balance are given in Eq. (2) While Eq. (3) represents external demand of finished goods. Eq. (4) guarantees that a set-up cost is incurred whenever a batch is purchased or produced. Finally, the back orders are not allowed and the production is either positive or zero in non-negative constraints Eq. (5).

4.4 Mathematical Formulation of CCMIMLLS Problem

The LSP considered in this paper can be explained as following:

There are N number of items for production in 'T' periods. The planning is to be done in such a way so as to meet the forecasted demand. In a production system, which is multistage, planning of each item is dependent on production of lower level items. Also, the capacity of the resources is finite i.e. there is a limitation on the resources available for production/ set up. The lead time is taken as zero.

Let us assume that there are N number of items to be planned in T periods (in the planning horizon). C_{it}, is the cost of production of 1 unit of item I in period t, h_{it} is the holding cost of 1 unit of item I in period t,, S_{it} is the setup cost of item i in period t, d_{it} is the demand for item I in period t, V_{ikt} is the amount of resource k necessary to produce item i in period t, b_{kt} is the amount of resource k available in period t, M is the upper bound on X_{it} ,S(i) the set of immediate successor items to item I, and r_{ij} is the number of units of item i needed by one unit of item j, where jϵ S(i).

Decision variables; x_{ij} is the lot size of item i in period t, y_{it} is '1' if item is produced in period t and zero otherwise. I_{it} the inventory of item I in period t.

$$\text{Min } (f(x))=\sum_{i=1}^{N} \sum_{t=1}^{T}(C_{it}X_{it} + h_{it}I_{it} + S_{it}Y_{it}) \quad \text{.....(1)}$$

$$I_{i,t-1} + X_{it} - I_{it} = d_{it} + \sum_{j\epsilon S(i)} r_{ij} X_{jt} \quad \ldots\ldots (2)$$

i=1, 2,…….N; T=1,2,……T

$$\sum_{i=1}^{N} (V_{ikt} X_{it} + f_{ikt} y_{it}) \leq b_{kt} \quad \ldots\ldots (3)$$

k=1, 2, 3,…….K; t=1,2,3,…….T

$$X_{it} \leq M y_{it} \qquad i=1,\ldots..N; \quad t=1,\ldots..,T \quad \ldots\ldots (4)$$

$$X_{it}, I_{it} \geq 0 \qquad i=1,\ldots..N; \quad t=1,\ldots..,T \quad \ldots\ldots (5)$$

$$y_{it} \in \{0,1\} \qquad i=1,\ldots..N; \quad t=1,\ldots..,T \quad \ldots\ldots (6)$$

The objective function (1) is to minimize the aggregate of production, inventory holding and setup costs in T periods. Equation (2) is inventory balance constraints, which establishes the association between inventory and production at the beginning and the end of the period. Constraints (3) shows the capacity limitations pertaining to production and setup. Constraint (4) ensures that the solution will have setup when it has production. Constraints which are at the last i.e. (5) and (6) require that variables must be positive and setup variables must be binary.

Combination of several factors like ordering cost, holding cost, shortage cost, capacity constraints, minimum and maximum order quantity etc. result in different models to be analyzed like capacitated or uncapacitated, single level or multi-level, single item or multi item models. Simple single product structures can be solved easily using mathematical equations . Since, CMIMLLS problems are having very large solution space they are considered as NP-hard problems which do not have solution with polynomial time. Therefore, soft computing techniques are necessary to compute optimum values of lot sizes.

4.5. Binary Particle Swarm Optimization (BPSO)

4.5.1. BPSO approach for Lot Sizing Problem

(a) Initial solution representation

Solution representation of particle p, X^{pk}_{id} , for BPSO is given in Table 4.1. This representation is due to Hernández and Suer (1999). Where each swarm contains ‘P’ number of particles referring to d dimensions and ‘i’ items. Here, ‘k’ represents iteration number.

A population of binary values (0 or 1) are randomly assigned for‘t’ dimensions of ‘i’ items in the MLLS problem for all ‘p’ particles which gives the information about where setups are made.

If $R_{id} > 0.5$ then $X_{id}=1$

else $X_{id}=0$;

R_{id}=random value.

i= item number=1, 2, 3…n

k=iteration number=1, 2, 3…..k

d=period=1, 2, 3……t

For initial generation k=0, i.e. $X_{id} = X^0_{id}$

	1	2	3	4	5	……	12
X^{pk}_{1d}	1	0	1	1	0	……	1
X^{pk}_{2d}	-	-	-	-	-	……	-
X^{pk}_{3d}	-	-	-	-	-	-	-
X^{K}_{id}	-	-	-	-	-	-	-

Table 4.1 Representation of Particle

Lot size:

According to particle solution lot sizes are calculated as shown in Table 4.2. The time periods where demand is not “0”, there set up has been made. Therefore, order quantity in particular period may be (i) requirement of that period or (ii) requirement of that period including with group of requirements of periods ahead or (iii) zero.

	1	2	3	4	5	……	12
L^{pk}_{1d}	140	0	155	175	0	……	
L^{pk}_{2d}	-	-	-	-	-	-	-
L^{pk}_{3d}	-	-	-	-	-	-	-
.	-	-	-	-	-	-	-

Table 4.2 Lot size according to particle dimension

L^{K}_{id} = lot size of item i ordered in period d at iteration k of particle p.

(b) Velocity of initial generation particles

After assigning particle dimensions, velocity values need to be calculated as shown in Table 4.3, to find next generation population. This velocity calculation is of 2 types i.e. 1) velocity calculation for initial generation (2) Velocity calculations for remaining generations.

Velocity values are restricted to some minimum and maximum namely

$V^{pk}_{id} = [V_{mini}, Vmaxi] = [-5, 5]$.

V^{pk}_{id} =velocity of particle of period d at iteration k

For initial generation, velocity values are calculated using following formula

$$\mathbf{V^{0p}_{id} = V_{mini} + (V_{maxi} - V_{mini}) * R}$$

R=a random value within 0 to 1, which is generated using rand ().

	1	2	3	4	5	……	12
V^{pk}_{1d}	-1.8	3.7	2.9	-0.69	-3.1	……	1.2
V^{pk}_{2d}	-	-	-	-	-	-	-
V^{pk}_{3d}	-	-	-	-	-	-	-
V^{pk}_{id}	-	-	-	-	-	-	-

Table 4.3 Velocity matrix of particle

(c) Particle best and global best

Particle having best fitness value [$f(x^k_p)$] is assigned to global best. As it is the initial generation all particle best values are equal to particle values as shown in Table 4.4.

Table 4.4. Particle Best and Global Best matrices

	1	2	3	4	5	……	12
PB^{pk}	1	0	1	1	0	……	1
PB^{pk}	-	-	-	-	-	……	-
PB^{pk}	-	-	-	-	-	-	-
PB^{pk}_{id}	-	-	-	-	-	-	-

	1	2	3	4	5	…	12
GB^k	1	0	1	1	0	…	1

(d) Updating parameters for next generations

(i) Updating velocity (V^{pk}_{id}):

(I) new velocity = $V^{pk}_{id} = P\,(V^{p,\,k-1}_{id} + \Delta V^{p,\,k-1}_{id})$

$\Delta V^{p,\,k-1}_{id} = c1\ R1\ (PB^{p,k-1}_{id} - X^{p,k-1}_{id}) + c2\ R2(GB^{K-1}_{id} - X^{p,k-1}_{id})$

c_1, c_2 are social and cognitive parameters, R_1 & R_2 are uniform random numbers between (0, 1)

Here Piece wise linear function [$P\,(V^{pk}_{id})$]

$P\,(V^{pk}_{id}) = Vmaxi$ if $V^{pk}_{id} > Vmaxi$

$= V^{pk}_{id}$ if $|V^{pk}_{id}|$ V_{maxi}

$= Vmini$ if $V^{pk}_{id} < Vmini$

(ii) Updating position (X^{pk}_{id}) by sigmoid function:

$X^{K}_{id} = 1$ if $R < S\,(V^{pk}_{id})$

$= 0$ otherwise

Sigmoid function $S\,(V^{pk}_{id})$:

This function forces velocity values to be in the limits of '0' to '1'. It helps to update next generation X^{pk}_{id} values.

$$sigmoid\left(v^{k}_{id}\right) = \frac{1}{1+e^{-v^{k}_{id}}}\ .$$

(iii) Updating particle best and global best (PB^{pk}_{id}, GB^{K}_{id})

After each and every iteration, update particle best and global best values according to the fitness values of particles in the newly generated swarm.

(e) Termination:

If the number of iterations reaches a predetermined value, called maximum number of iterations then stop searching, otherwise go to (d).

4.5.2 PSO approach Pseudo Code

STEP1: Initialization phase

Initialize swarm

Assign velocities

Fitness calculation (2 objectives i.e. setup cost and holding cost)

Particle best and global best

STEP2: Iteration phase with PSO

for (i=0; i<number of iterations; i++)

{

Update velocity

Update dimension

Fitness calculation

Particle and global best

}

4.5.3 Numerical Example

In this example two items are there in which item-2 is having independent demand and the demand of the other item (i.e. item-1) depends on first one. Table 4.5 represents the demands and also depicts different costs involved in the problem. Fig. 4.1 represents BOM structure.

	1	2	3	4	5	6
Demand	20	100	50	30	10	100

Table 4.5. Product demand and setup and holding cost

Item	S.C.	H.C.
1	500	50
2	100	10

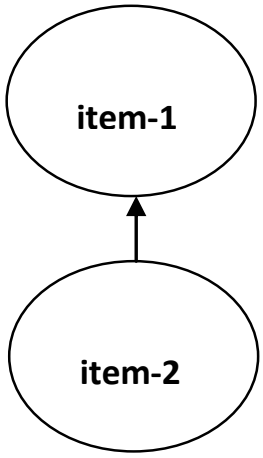


Figure 4.1. BOM structure for 2×6 problem

The problem is mapped and executed in terms of BPSO and the steps are given below:

Step1: initial generation

Particle 1

Item1	1	0	0	0	1	1
	200	0	0	0	10	100

	2.9	-1.8	3.5	-1.2	0.7	3.8
Item2	1	0	0	0	1	0
	200	0	0	0	110	0
	-1.5	1.6	-2.7	3.3	1.8	-3.9

Fitness fuction=$f(x^0_1)$=17200

Particle 2

Item1	1	1	1	0	0	1
	20	100	90	0	0	100
	-2.5	1.7	3	-4	1.6	2.3
Item2	1	0	0	0	0	1
	210	0	0	0	0	100
	2.2	-3.9	1.4	3	-4.6	2

Fitness function=$f(x^0_2)$=7500

Particle 3

Item1	1	0	1	0	1	1
	120	0	80	0	10	100
	3	-4.6	2	-1.2	0.7	3.8
Item2	1	0	1	0	1	0
	120	0	80	0	110	0
	-3	-4	1.6	2.9	-1.8	3.5

Fitness fuction=$f(x^0_2)$ =9800

Step 2: As it is initial generation, assign each particle in the swarm to particle best(PB)

$PB^{1,0}_{1,1}=X^{1,0}_{1,1}$, $PB^{1,0}_{1,2}=X^{1,0}_{1,2}$…….. $PB^{1,0}_{1,12}=X^{1,0}_{1,12}$

$PB^{1,0}_{2,1}=X^{1,0}_{2,1}$, $PB^{1,0}_{2,2}=X^{1,0}_{2,2}$…….. $PB^{1,0}_{2,12}=X^{1,0}_{2,12}$

	d	1	2	3	4	5	6	fitness
$PB^{1,0}_{id}$	i=1	1	0	0	0	1	1	17200
	i=2	1	0	0	0	1	0	
$PB^{2,0}_{id}$	i=1	1	1	1	0	0	1	7500
	i=2	1	0	0	0	0	1	
$PB^{3,0}_{id}$	i=1	1	0	1	0	1	1	9800
	i=2	1	0	1	0	1	0	
	D	1	2	3	4	5	6	fitness
GB_{id}^0	i=1	1	1	1	0	0	1	7500
	i=2	1	0	0	0	0	1	

Step 3: Updating velocity using piece wise function

Assume C1=C2=1, r1=r2=0.5;

Update particle dimension

$\Delta V^{1,0}{}_{12}$= c1 R1 ($PB^{1,0}{}_{12} - X^{1,0}{}_{12}$) + c2 R2($GB^{0}{}_{12} - X^{1,0}{}_{12}$)

$\Delta V^{1,0}{}_{12}$=1*0.5(0-0)+1*0.5(1-0)= 0.5

$V^{1,1}{}_{12}$=P ($V^{1,0}{}_{12}+\Delta V^{1,0}{}_{12}$)=P(-1.8+0.5)=-1.3

Updating particle position

R (0, 1)=0.11 < Sigmoid(-1.3)=0.21

So new dimension value= $X^{1}{}_{12}$=1

After completion of velocity calculations of all dimensions, particles are updated as follows

	D	1	2	3	4	5	6	fitness
$X^{1,1}{}_{id}$	i=1	1	1	1	1	0	1	6400
	$V^{1,1}{}_{1d}$	2.9	-1.3	4	-1.2	0.2	3.8	
	Sig($V^{1,1}{}_{1d}$)	0.94	0.21	0.98	0.23	0.54	0.97	
	Random	0.72	0.11	0.4	0.2	0.67	0.8	
	i=2	1	0	0	0	0	1	
	$V^{1,1}{}_{2d}$	-1.5	1.6	-2.7	3.3	1.3	3.3	
	Sig($V^{1,1}{}_{2d}$)	0.18	0.83	0.06	0.96	0.78	0.96	
	Random	0.10	0.91	0.10	0.99	0.80	0.91	
$X^{2,1}{}_{id}$	i=1	1	1	1	1	0	1	3500
	i=2	1	1	1	1	0	1	
$X^{3,1}{}_{id}$	i=1	1	0	0	0	1	1	9800
	i=2	1	0	0	0	1	0	

Updated particle best matrix

	D	1	2	3	4	5	6	fitness
$PB^{1,1}_{id}$	i=1	1	1	1	1	0	1	6400
	i=2	1	0	0	0	0	1	
$PB^{2,1}_{id}$	i=1	1	1	1	1	0	1	3500
	i=2	1	1	1	1	0	1	
$PB^{3,1}_{id}$	i=1	1	0	1	0	1	1	9800
	i=2	1	0	1	0	1	0	

Global best matrix

	D	1	2	3	4	5	6	fitness
GB_{id}^{1}	i=1	1	1	1	1	0	1	3500
	i=2	1	1	1	1	0	1	

Step 4: Termination

Repeat this procedure (step3) until iteration number k< max iteration.

4.6. Hybrid PSO approach for Lot sizing Problem

4.6.1 Hybrid PSO approach Pseudo Code

STEP1: Initialization phase

Initialize swarm

Assign velocities

Fitness calculation

Particle best and global best

STEP2: Iteration phase with PSO

for (i=0; i<number of iterations; i++)

{

Update velocity

Update dimension

Fitness calculation

Particle and global best

}

STEP3: Iteration phase with local search

for (i=0; i<number of iterations; i++)

{

Update velocity

Update dimension

Fitness

Iterative improvement local search

Particle and global best

}

STEP4: Iteration phase by local search for global best value

for (i=0; i<number of iterations; i++)

{

Iterative improvement local search}

Termination

4.6.2 Numerical Example for Hybrid PSO

For numerical example 7×6 problem is taken from Jinxing Xie, Jiefang Dong's Heuristic Algorithm For general Capacitated lot sizing problem (2002), this example is taken for the comparison with other problem considered in the paper.

M.Fatih Tasgetiren and Yun-Chia Liang (2003) state that if population size (number of particles in swarm) is at least double the number of periods in the planning horizon, performance would be better. According to Yuhui Shi (2004), PSO with minimum population size 5 gives better performance.

But for convenience swarm size i.e. population size is taken as 3 in numerical example, even though all the problems are solved with population size of 40.

Step1:

Swarm contains 3 particles, each of size 7×6

$$\text{Particle1} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} +4.3 & -2.4 & -3.8 & +2.5 & +4.3 & +1.5 \\ -3.6 & -1.4 & -1.4 & -0.9 & -3.6 & +4.4 \\ -1.2 & +0.6 & +0.2 & -1.4 & -1.2 & +3.2 \\ -2.0 & +0.5 & -3.5 & -1.9 & -2.0 & -0.9 \\ +3.6 & -2.5 & -3.9 & -1.3 & +3.6 & +4.0 \\ +4.3 & +0.0 & +1.2 & -1.6 & +4.3 & +0.3 \\ +1.2 & -3.4 & +0.2 & -4.2 & +1.2 & -1.7 \end{bmatrix} \rightarrow \text{Fitness} = 10948$$

$$\text{Particle2} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} +1.5 & +3.4 & -1.8 & +2.5 & +4.3 & +1.5 \\ -0.6 & -1.4 & -1.4 & -0.9 & -3.6 & +4.4 \\ +2.5 & +1.6 & +0.2 & -1.4 & -1.2 & -3.2 \\ -1.1 & +0.5 & -3.5 & -0.9 & +2.0 & +0.9 \\ +2.2 & -2.5 & -3.9 & -1.3 & +3.6 & +4.0 \\ -1.3 & +0.0 & +1.2 & -1.6 & +4.3 & +0.3 \\ +0.7 & -3.4 & +0.2 & -4.2 & +1.2 & -1.7 \end{bmatrix} \rightarrow \text{Fitness} = 11648$$

$$\text{Particle3} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} +0.5 & +5.4 & -1.8 & +2.5 & +3.3 & +2.5 \\ -0.1 & -2.4 & -1.4 & -0.9 & -3.6 & +1.4 \\ +2.5 & +0.6 & +1.2 & -1.4 & -3.2 & -3.2 \\ -3.1 & +0.5 & -3.5 & -1.9 & +2.0 & +0.9 \\ +2.2 & -2.5 & -3.9 & -1.3 & +3.6 & +4.0 \\ -1.3 & +0.0 & +2.2 & -1.6 & +4.3 & +0.3 \\ +0.7 & -3.4 & +0.2 & -4.2 & +1.2 & -1.7 \end{bmatrix} \rightarrow \text{Fitness} = 9376$$

Step2:

As it is first generation assign all particle values to particle best, and best fitness particle dimensions to global best value

$$\text{PB}_1 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad \text{PB}_2 \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{PB}_3 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\text{Global Best} = \text{GB} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Step3:

Update Velocity using standard procedure of Binary particle swarm optimization

$$\text{Particle1} = \begin{bmatrix} +4.3 & -2.4 & -3.8 & +2.5 & +4.3 & +0.04 \\ -3.6 & -1.4 & -1.4 & -0.9 & -3.6 & +4.4 \\ -1.2 & +0.6 & +0.2 & -1.4 & -1.2 & +1.74 \\ -2.0 & +0.5 & -3.5 & -1.9 & -2.0 & -0.9 \\ +3.6 & -2.5 & -3.9 & -1.3 & +3.6 & +4.0 \\ +4.3 & +0.0 & +1.2 & -1.6 & +4.3 & -1.16 \\ +1.2 & -4.86 & +1.66 & -4.2 & +1.2 & -3.16 \end{bmatrix} \rightarrow \begin{bmatrix} 0.98 & 0.08 & 0.02 & 0.92 & 0.98 & 0.50 \\ 0.02 & 0.19 & 0.19 & 0.28 & 0.02 & 0.98 \\ 0.23 & 0.64 & 0.54 & 0.19 & 0.23 & 0.85 \\ 0.11 & 0.62 & 0.02 & 0.13 & 0.11 & 0.28 \\ 0.97 & 0.07 & 0.01 & 0.21 & 0.97 & 0.98 \\ 0.98 & 0.50 & 0.76 & 0.16 & 0.98 & 0.23 \\ 0.76 & 0.00 & 0.84 & 0.01 & 0.76 & 0.04 \end{bmatrix}$$

$$\text{Sigmoid}(V+\Delta V)=\begin{bmatrix}0.98 & 0.08 & 0.02 & 0.92 & 0.98 & 0.50\\ 0.02 & 0.19 & 0.19 & 0.28 & 0.02 & 0.98\\ 0.23 & 0.64 & 0.54 & 0.19 & 0.23 & 0.85\\ 0.11 & 0.62 & 0.02 & 0.13 & 0.11 & 0.28\\ 0.97 & 0.07 & 0.01 & 0.21 & 0.97 & 0.98\\ 0.98 & 0.50 & 0.76 & 0.16 & 0.98 & 0.23\\ 0.76 & 0.00 & 0.84 & 0.01 & 0.76 & 0.04\end{bmatrix} \text{R}=\begin{bmatrix}0.0 & 0.5 & 0.09 & 0.90 & 0.99 & 0.71\\ 0.0 & 0.3 & 0.99 & 0.11 & 0.33 & 0.99\\ 0.0 & 0.72 & 0.81 & 0.89 & 0.54 & 0.89\\ 0.0 & 0.92 & 0.00 & 0.93 & 0.33 & 0.37\\ 0.0 & 0.10 & 0.80 & 0.10 & 0.98 & 0.99\\ 0.0 & 0.65 & 0.84 & 0.97 & 0.99 & 0.37\\ 0.0 & 0.5 & 0.98 & 0.70 & 0.85 & 0.35\end{bmatrix}$$

$$\text{New Particle1}=\begin{bmatrix}1 & 0 & 0 & 1 & 0 & 0\\ 1 & 0 & 0 & 1 & 0 & 0\\ 1 & 0 & 0 & 0 & 0 & 0\\ 1 & 0 & 1 & 0 & 0 & 0\\ 1 & 0 & 0 & 1 & 0 & 0\\ 1 & 0 & 0 & 0 & 0 & 0\\ 1 & 0 & 0 & 0 & 0 & 0\end{bmatrix}=\rightarrow \text{Fitness}=10300$$

As particle 1 fitness value is improved, so first particles, (particle best (PB_1)) value will be updated with current particle data. If fitness is not improved, then particle best value will remain same.

Like this, particle best and global best values will be updated for all particles according to fitness values.

Step4: Repeat this procedure until iteration number k < max iteration.

Local Search: Local Search:

$$\text{input Particle}=\begin{bmatrix}1 & 0 & 0 & 1 & 0 & 0\\ 1 & 0 & 0 & 1 & 0 & 0\\ 1 & 0 & 0 & 0 & 0 & 0\\ 1 & 0 & 1 & 0 & 0 & 0\\ 1 & 0 & 0 & 1 & 0 & 0\\ 1 & 0 & 0 & 0 & 0 & 0\\ 1 & 0 & 0 & 0 & 0 & 0\end{bmatrix}\rightarrow \text{new paticle}=\begin{bmatrix}1 & 0 & 1 & 0 & 0 & 0\\ 1 & 0 & 0 & 1 & 0 & 0\\ 1 & 0 & 0 & 0 & 0 & 0\\ 1 & 0 & 1 & 0 & 0 & 0\\ 1 & 0 & 0 & 1 & 0 & 0\\ 1 & 0 & 0 & 0 & 0 & 0\\ 1 & 0 & 0 & 0 & 0 & 0\end{bmatrix}\rightarrow \text{Fitness}=9820$$

As solution is improved, old particle (i.e. input particle) should be replaced with new particle.

4.7. Single item Uncapacitated Lot Sizing Problem (1×12)

Here Table 4.6 shows the Demand of end product and costs involved in 1×12 problem and Table 4.7 shows the comparison of results for the same problem

period	1	2	3	4	5	6	7	8	9	10	11	12
Demand	15	5	15	110	65	165	125	25	90	15	140	115

Table 4.6 Demand of end product and costs involved in 1×12 problem

ITEM No.	1×12 problem	
	H.C	S.C
1	97.83	780

Comparison of results with Lot for Lot Solution

	LFL	BGA	HBGA	BPSO	HBPSO
1×12	9360	9069.15	9069.15	9069.15	9069.15

Table 4.7. 1×12 problem solution with BGA, HBGA, BPSO and HBPSO

4.8. Single-item, multi-level Capacitated Lot Sizing Problem (7×6)

Figure 4.2 shows the BOM structure of 7×6 problem. Table 4.8 shows the Demand and availability of end product and costs involved in 7×6 problem and Table 4.9 shows the comparison of results for the same problem

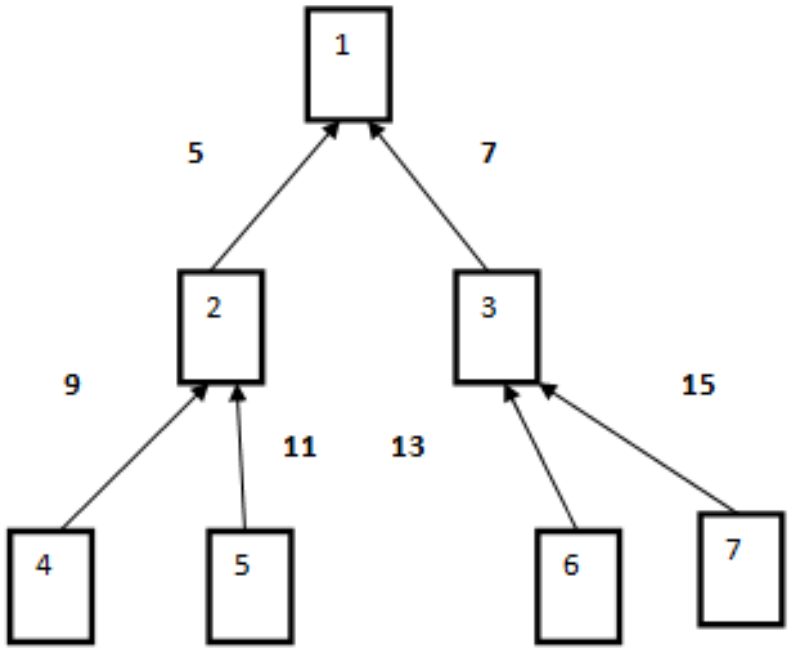


Figure 4.2. BOM Structure of 7×6 problem

Item No.	1	2	3	4	5	6	7
H.C.	12	0.6	1	0.04	0.03	0.04	0.04
S.C.	400	500	1000	300	200	400	100

Item No.	1	2	3	4	5	6
demand	40	0	100	0	90	10
Available capacity	10000	0	5000	5000	1000	1000

Table 4.8 Demand and availability of end product and different costs involved in 7×6 problem

	BGA total cost	HBGA total cost	% of improvement	BPSO total cost	% of improvement	HBPSO total cost	% of improvement
7×6	9245	8320	10	8320	10	8320	10

Table 4.9 **7**×6 problem solution with BGA,HBGA ,BPSO and HBPSO

4.9. Single-item, multi-level Capacitated Lot Sizing Problem (50×12)

Figure 4.3 shows the BOM structure of 50×12 problem, Table 4.10 shows the Demand and availability of end product and costs involved in 50×12 problem. Table 4.11 and Figure 4.4 shows the comparison of results for the same problem at different iterations.

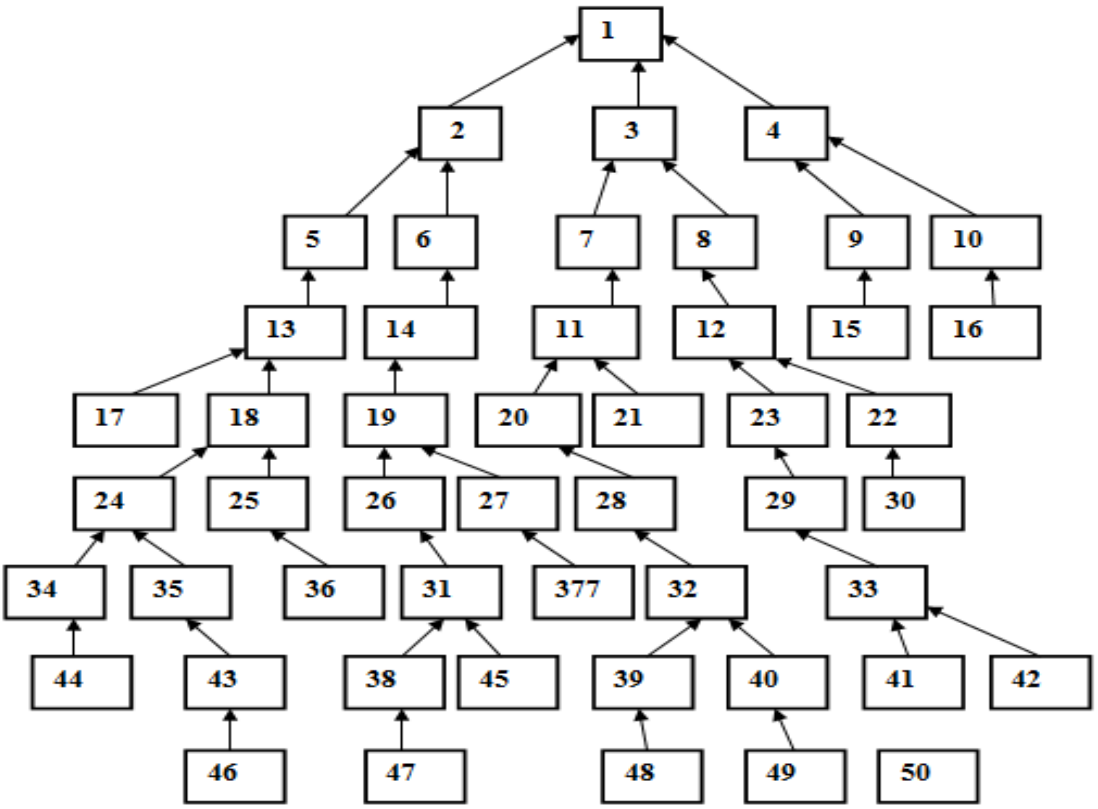


Figure 4.3 BOM Structure of 50×12 problem

S.No	50×12 problem	
	H.C	S.C
1	97.83	780
2	45.19	200
3	43.82	590
4	5.82	710
5	26.04	890
6	18.87	610
7	27.03	920
8	15.64	210
9	2.67	490
10	1.86	920
11	23.5	520
12	12.59	540
13	25.13	510
14	16.42	500
15	0.84	300

16	1.02	450
17	0.62	440
18	23.71	510
19	15.32	910
20	20.58	830
21	8.71	730
22	3.14	850
23	0.94	450
24	13.02	370
25	7.34	390
26	7.53	540
27	4.36	160
28	18.52	480
29	5.81	410
30	1.93	140
31	6.71	390
32	15.35	370
33	4.36	520
34	3.28	700
35	6.38	160
36	3.47	290
37	1.97	420
38	1.76	160
39	6.41	450
40	7.17	340
41	2.97	750
42	0.25	140
43	3.22	430
44	1.85	890
45	3.84	610
46	0.41	860
47	0.37	860
48	3.84	350
49	3.95	610
50	1.63	350

period	1	2	3	4	5	6	7	8	9	10	11	12
Demand	15	5	15	110	65	165	125	25	90	15	140	115
Available	1000	2000	1000	0	5000	1000	0	500	800	500	1000	200

Table 4.10 Demand and availability of end products and different costs involved in 50×12 problem

50×12				
Iteration No.	**PSO**	**HPSO**	**BPSO**	**HBPSO**
5	386,785.09	380,765.30	280,295.00	250295.00
25	380,891.31	352114.59	243,797.00	241009.15
50	350,503.75	330138.87	203,956.09	200037.17
100	322,136.16	321142.15	193,128.11	199121.89
200	279,484.72	290477.29	192,017.59	195192.04
500	249,875.41	250132.65	189,013.95	185013.09
1,000	234,587.08	230513.19	186,579.11	182599.11
2,000	234,587.08	232187.12	186,543.84	183450.08
5,000	234,489.03	223154.89	185,042.16	174057.32
10,000	229,484.6	219803.29	184,629.19	173753.29
15,000	229,484.6	214040.12	181,685.31	173753.29
20,000	204,240.90	213108.00	181,685.31	173753.29
30,000	204,140.90	191617.40	181,685.31	173753.29

Table 4.11 50×12 problem solution with PSO, HPSO, BPSO and HBPSO

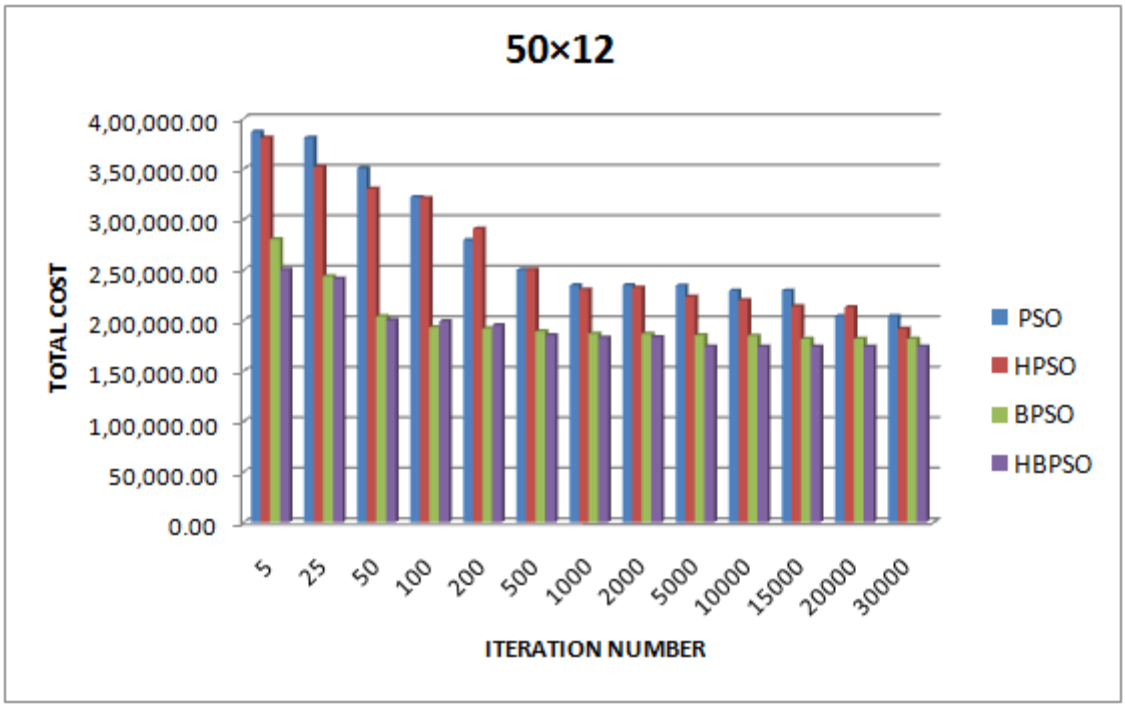


Figure 4.4 50×12 problem solution at different iterations with PSO, HPSO, BPSO and HBPSO

Figure 4.5 Convergence Graph of 50×12 problem.

4.10. Multi-item, multi-level Capacitated Lot Sizing Problem (39×12)

Figure 4.6 shows the BOM structure of 39×12 problem, Table 4.12 shows the Demand and availability of end product and costs involved in 39×12 problem. Table 4.15 and Figure 4.8 shows the comparison of results for the same problem.

Table 4.14 and Figure 4.7 give the information regarding swarm size (i.e. number of particles in swarm) effect on Fitness at different iterations of problem. Table 4.13 shows the particle average values comparison at different iterations.

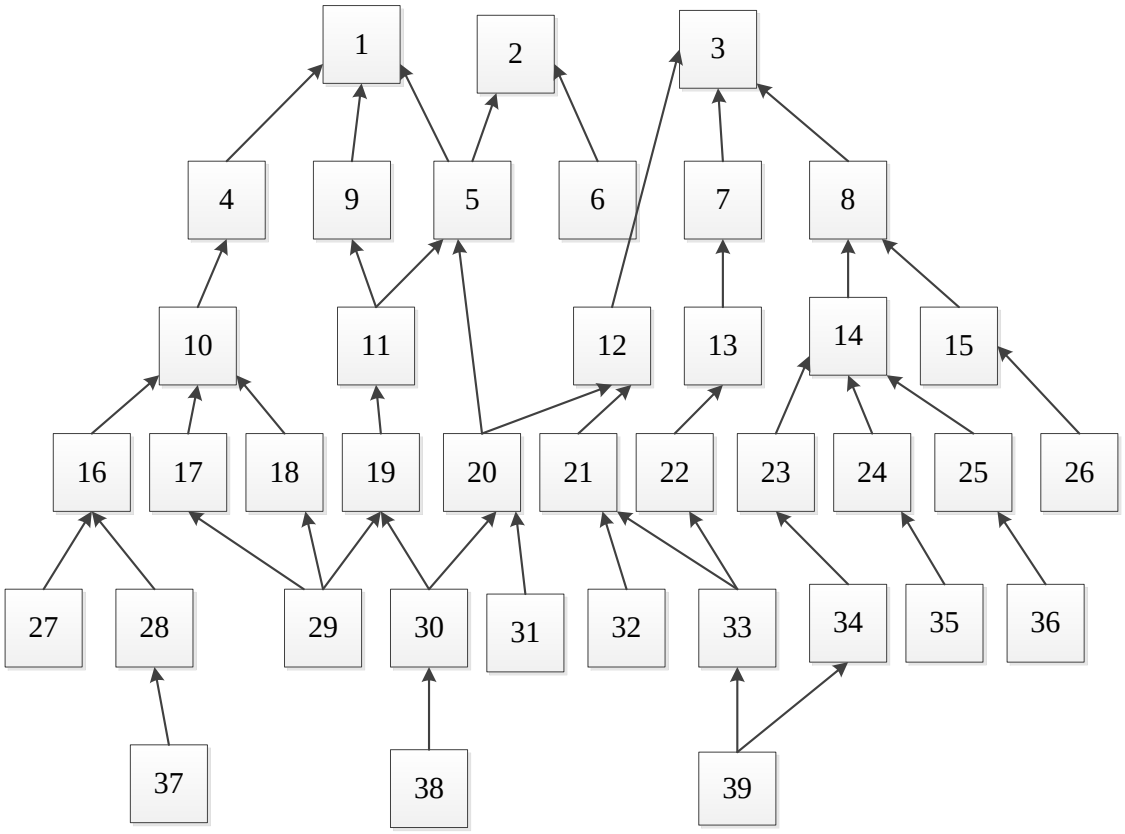


Figure 4.6 BOM Structure of 39×12 problem

	39*12 problem	
	H.C	S.C
1	40.08	490
2	35.27	450
3	59.66	90
4	25.42	140
5	10.42	880
6	22.64	440
7	22.31	70
8	19.53	430
9	1.34	930
10	25.12	650
11	9.46	740
12	17.48	680
13	4.32	800
14	14.28	220
15	2.56	850
16	10.07	400
17	4.59	650
18	7.13	860
19	8.82	850
20	10.6	670
21	6.02	370
22	2.78	360
23	2.95	310
24	9.32	440
25	0.31	590
26	1.45	580
27	3.63	650
28	4.35	450
29	3.29	820
30	5.04	620
31	2.53	580
32	3.3	340
33	0.61	340
34	2.52	80
35	4.83	690
36	3.44	430
37	0.91	60
38	2.64	760
39	2.65	180

period	1	2	3	4	5	6	7	8	9	10	11	12
Item1	10	100	10	130	115	150	70	10	65	70	165	125
available	1500	2000	0	1000	800	5000	0	800	500	1000	2000	200
Item2	175	15	85	90	85	90	75	150	75	10	150	15
available	0	1000	2000	1000	900	0	800	1200	500	500	1000	100
Item3	135	165	15	105	25	120	50	60	5	140	60	10
available	1000	2000	900	800	0	1000	1200	300	500	800	100	100

Table 4.12 Demand and availability of end products and different costs involved in 39×12 problem

Paticle No	300^{th} iteration	400^{th} iteration	500^{th} iteration	600^{th} iteration
1	2,87,163.81	2,96,827.84	2,71,354.18	2,52,984.90
2	3,36,654.96	3,67,410.31	2,89,081.71	3,15,492.96
3	2,93,118.93	2,69,643.84	2,45,523.12	2,84,151.09
4	3,06,763.81	3,02,162.75	2,62,019.62	2,43,960.09
5	3,45,834.90	2,48,838.14	3,14,560.21	3,29,510.21
6	2,82,012.65	2,66,686.15	2,56,661.42	2,81,553.00
7	3,15,013.18	2,72,906.25	2,61,492.73	3,52,466.06
8	2,96,122.84	3,48,605.81	2,83,168.21	2,88,089.43
9	2,48,314.07	3,01,639.96	3,05,530.84	2,54,905.98
10	2,47,896.62	2,57,145.29	3,03,904.59	2,43,397.90
11	2,77,441.78	2,69,856.78	3,84,417.81	2,92,368.90
12	3,62,870.81	3,06,533.00	2,76,282.09	2,98,579.78
13	3,21,958.31	3,71,249.90	2,94,672.75	2,75,554.75
14	2,78,471.87	2,71,883.00	2,74,612.28	2,96,049.25
15	2,60,861.73	3,11,140.62	2,97,362.34	2,72,623.87
16	2,85,929.62	2,86,662.00	2,69,972.37	2,99,126.06
17	2,63,018.00	2,89,455.09	3,51,597.21	2,66,942.46
18	2,78,906.21	2,64,557.53	2,84,343.62	2,67,243.21
19	3,02,362.43	2,98,466.21	2,85,951.65	2,53,033.96
20	3,14,437.37	2,74,489.31	2,95,188.09	2,40,238.76
21	3,04,807.06	3,04,971.06	3,92,710.34	2,66,432.03
22	2,59,932.62	2,61,123.39	3,28,282.68	2,84,568.34
23	2,71,942.18	2,59,024.79	2,61,638.71	3,57,821.31
24	2,32,535.56	2,67,100.62	3,26,993.50	2,94,543.37
25	3,13,567.40	2,59,005.15	2,80,591.90	3,06,715.31
26	2,96,707.34	2,68,255.06	2,43,848.54	3,58,286.62
27	2,77,164.31	2,81,731.18	2,64,352.62	2,76,793.18
28	2,61,792.07	2,74,739.06	2,89,422.71	2,99,171.06
29	3,80,921.25	2,56,552.51	2,77,678.18	2,58,015.42

30	2,92,137.40	2,66,886.78	2,79,697.71	2,56,473.42
31	2,71,272.21	2,80,590.34	2,87,545.06	3,19,100.50
32	2,49,626.34	2,65,390.56	3,28,519.46	2,62,788.46
33	3,06,842.96	2,51,897.54	3,50,235.21	3,15,292.06
34	2,64,361.06	2,94,248.43	2,47,796.82	2,41,895.12
35	2,69,949.12	3,26,534.81	3,37,587.68	2,48,329.37
36	3,21,372.84	2,48,451.96	2,43,114.26	2,41,233.20
37	2,95,246.15	2,61,236.87	2,56,835.37	2,19,248.84
38	2,72,370.71	2,96,141.87	3,38,252.03	3,23,139.43
39	2,75,238.12	3,46,146.93	2,58,783.17	2,85,475.75
40	3,26,699.65	2,59,180.64	2,81,134.03	2,83,344.40
Avg	2,91,241.00	2,85,134.20	2,88,971.50	2,82,673.50

Table 4.13 Particle average values comparison at different iterations

Itr.No	SWARM10	SWARM20	SWARM30	SWARM40
5000	203257	198162	193931	193219
10000	195932	198162	193931	185691
50000	195930	193385	191943	175684
100000	195669	180456	184664	172682
150000	191101	180456	184664	172682
200000	190074	180456	180456	172682

Table 4.14 Swarm Size Effect on Fitness

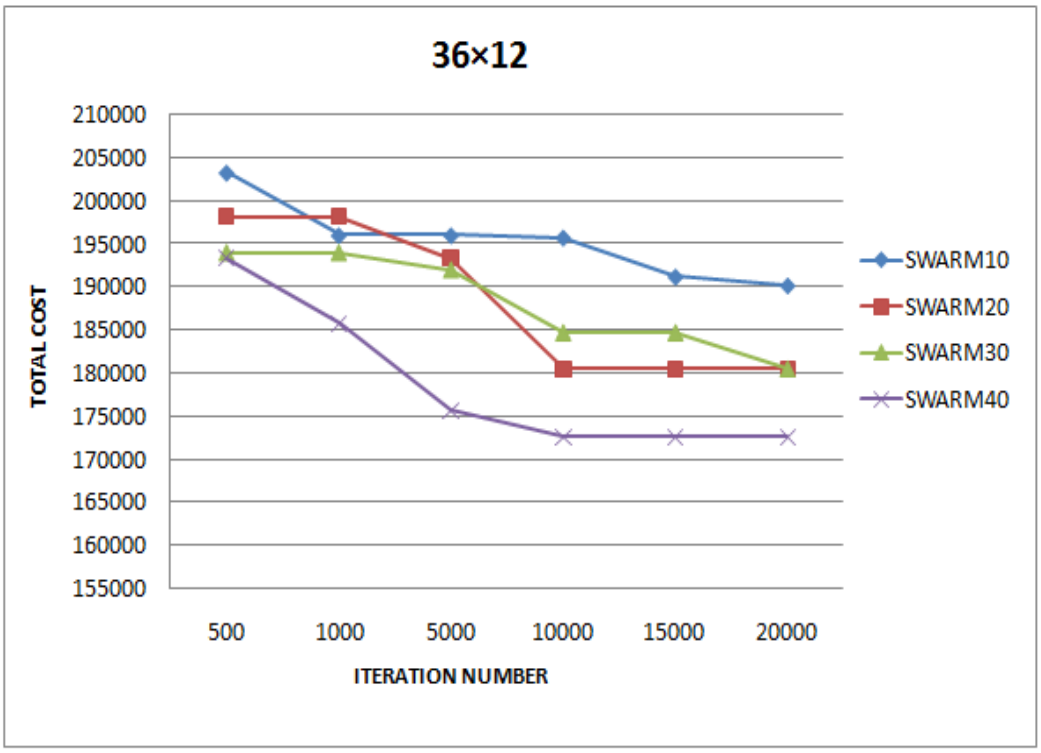


Figure 4.7 Swarm Size Effect on Fitness

39×12				
Iteration	**PSO**	**HPSO**	**BPSO**	**IIBPSO**
5	377,421.19	350605.65	246,901.17	246,901.17
25	327,867.12	239426.79	217,583.65	213605.76
50	242,463.20	204744.77	204,084.98	197578.04
100	221,525.29	178650.31	202,884.17	191770.14
200	199,022.79	178346.06	194,724.84	191770.14
500	197,410.34	178244.00	193,219.70	186117.70
1,000	197,410.34	177609.65	185,691.15	142889.60
2,000	197,410.34	177609.65	185,691.15	142889.60
5,000	197,410.34	177609.65	172,684.78	142889.60
10,000	197,410.34	177609.65	172,682.56	142889.60
15,000	197,410.34	177609.65	172,682.56	142889.60

Table 4.15 39×12 problem solution with PSO, HPSO, BPSO and HBPSO

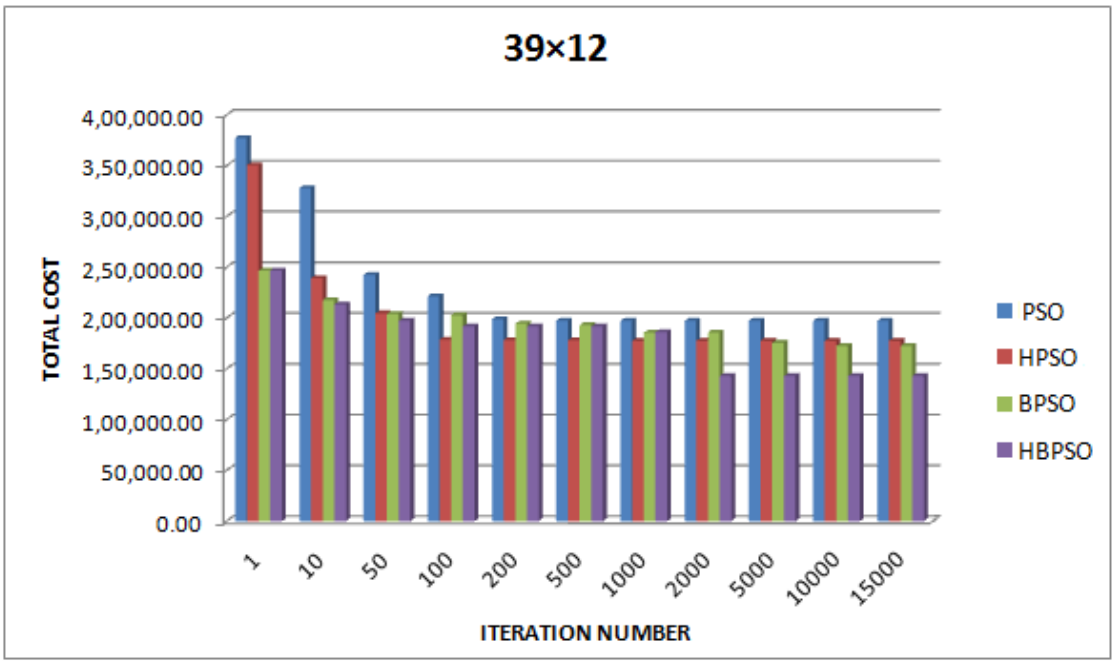


Figure 4.8 39×12 problem solution at different iterations with PSO, HPSO, BPSO and HBPSO

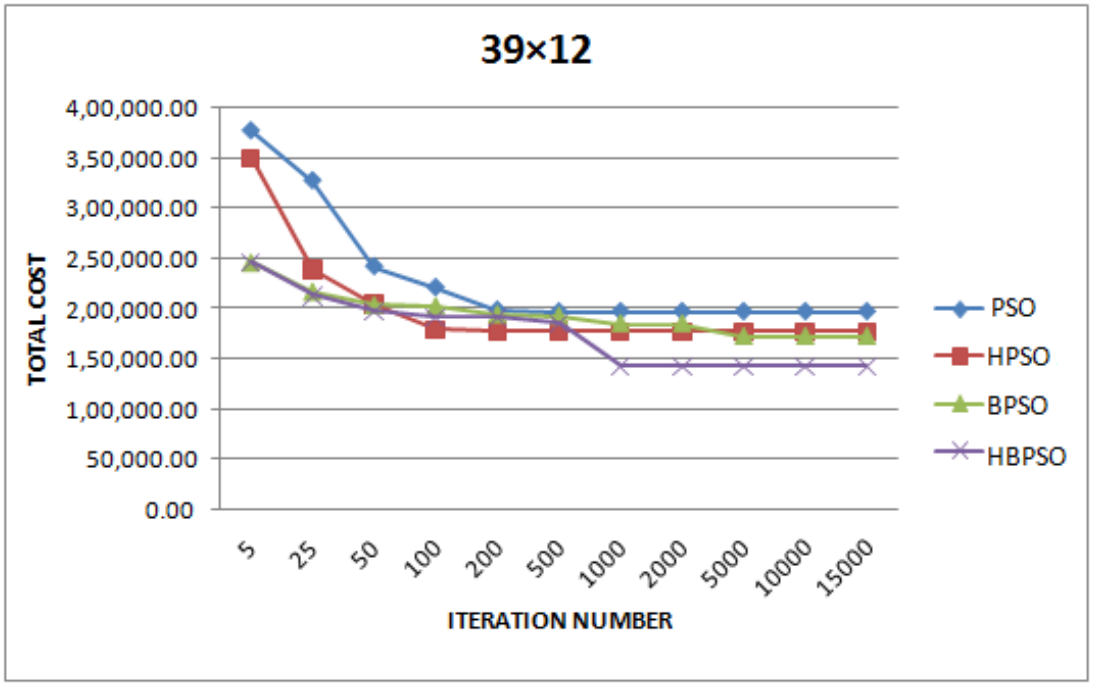


Figure 4.9 Convergence Graph of 39×12 problem

4.11. Multi-Item, Multi-Level Capacitated Lot Sizing Problem (75×36)

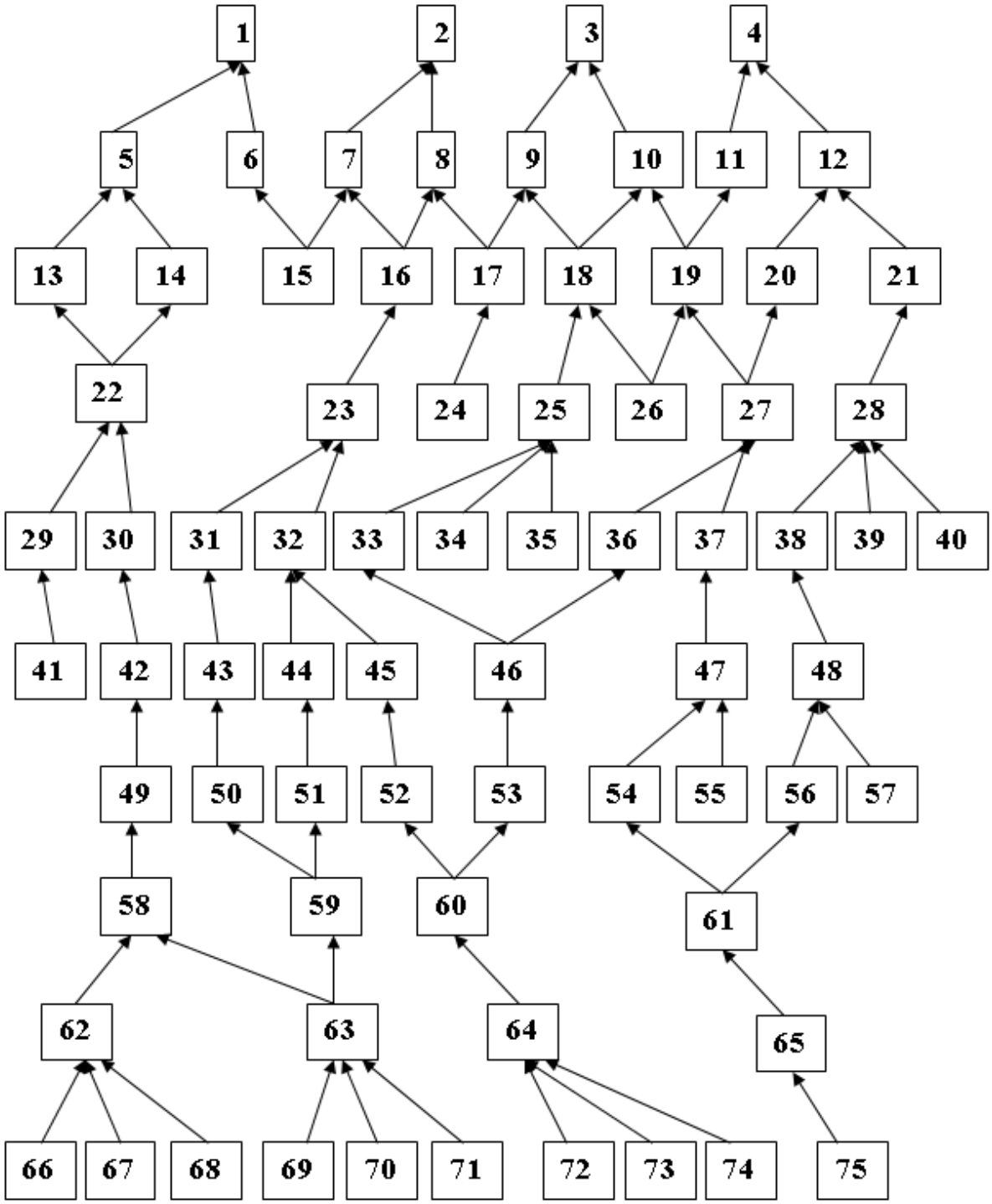


Figure 4.10 BOM Structure of 75×36 problem

S.No	75*36 problem	
	H.C	S.C
1	50	410
2	49	450
3	50	430
4	48	420
5	47.2	250
6	46	300
7	42	500
8	42.5	800
9	40	400
10	40.5	500
11	37	200
12	36	330
13	45	480
14	40	450
15	37	380
16	40	200
17	36	100
18	35	100
19	35	120
20	34	280
21	33	270
22	35	290
23	35	320
24	33	380
25	30	560
26	30	580
27	31	620
28	30	610
29	30	490
30	30	300
31	29	200
32	29	200
33	25	100
34	25	120
35	25	300
36	27	400
37	27	200
38	25	800
39	25	100
40	25	250
41	27	450
42	28	100
43	26	200
44	25	800
45	26	100

46	24	500
47	24	480
48	22	250
49	21	600
50	19	100
51	18	800
52	17	410
53	16	350
54	15	320
55	14	280
56	13	280
57	12	180
58	11	680
59	10	190
60	9	100
61	8	480
62	7	200
63	6	270
64	5	600
65	4	210
66	3	700
67	3	100
68	3	200
69	3	100
70	3	150
71	3	200
72	2	100
73	2	200
74	2	100
75	1	100

period	1	2	3	4	5	6	7	8	9	10	11	12
Item1	10	100	10	10	70	10	20	10	10	50	10	70
available	∞	∞	∞	∞	0	∞	∞	∞	∞	∞	∞	∞
Item2	20	10	10	10	100	20	10	10	10	320	10	100
available	∞	∞	0	∞	5000	∞	∞	∞	∞	∞	∞	∞
Item3	30	10	10	100	10	10	20	10	40	100	10	10
available	∞	∞	∞	∞	∞	5000	∞	∞	∞	∞	∞	∞
Item4	40	10	10	30	10	10	10	10	100	10	10	120
available	∞	∞	∞	0	∞	∞	∞	∞	∞	∞	∞	∞
period	13	14	15	16	17	18	19	20	21	22	23	24
Item1	10	100	10	60	10	10	50	10	10	10	30	10
available	∞	∞	∞	∞	∞	0	∞	∞	∞	∞	∞	∞
Item2	10	20	10	170	10	10	50	10	10	10	210	10
available	∞	∞	0	∞	∞	∞	∞	∞	∞	∞	∞	∞
Item3	10	180	10	10	10	10	60	10	10	10	10	10
available	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

Item4	10	10	10	110	10	10	30	10	410	10	20	10
available	∞	∞	∞	0	∞	∞	∞	∞	∞	∞	∞	∞
period	25	26	27	28	29	30	31	32	33	34	35	36
Item1	20	10	90	10	10	310	10	250	10	10	90	10
available	∞	∞	∞	∞	0	∞	∞	∞	∞	1000	∞	∞
Item2	10	10	10	1000	10	10	10	10	10	10	80	10
available	∞	∞	∞	∞	800	0	∞	∞	500	∞	∞	∞
Item3	600	10	100	10	10	10	10	10	600	10	10	10
available	∞	∞	∞	∞	0	∞	∞	∞	0	∞	∞	∞
Item4	50	10	10	10	800	10	10	10	90	10	10	10
available	∞	∞	∞	∞	0	∞	1000	∞	∞	∞	∞	∞

Table 4.16 Demand, availability of end product and different costs involved in 75×36 problem

75×36				
Iter No.	**PSO**	**HPSO**	**BPSO**	**HBPSO**
5	152174144	151074134	89866320	89866320
25	145240592	143150594	86317160	80226251
50	131999600	128899511	79341128	77312117
100	108485416	106374426	71873080	61752171
200	99614824	99919883	60409328	60409328
500	89866320	99614824	50344516	47817140
1,000	86317160	54844216	47819130	39071648
5,000	65511652	50344516	43816120	36459912
10,000	54344516	50344516	43816120	36291480
20,000	54344516	47444516	41817140	36205080
30,000	54344516	47344516	41817140	36205080

Table 4.17 75×36 problem solution with PSO, HPSO, BPSO and HBPSO

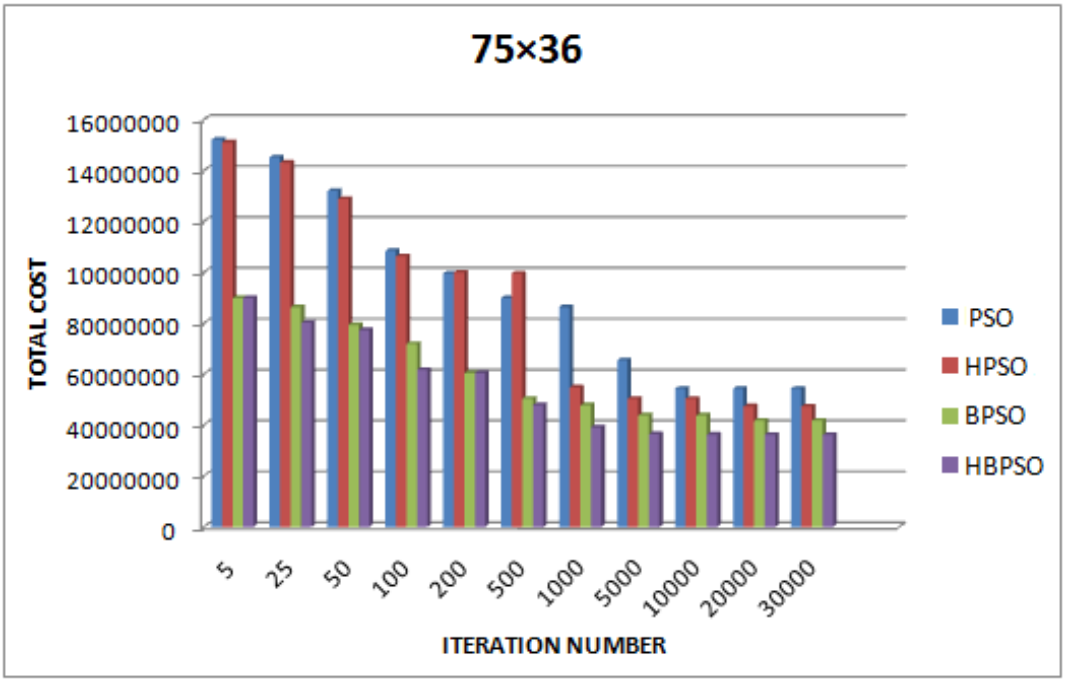


Figure 4.11 75×36 problem solutions at different iterations with PSO, HPSO, BPSO and HBPSO

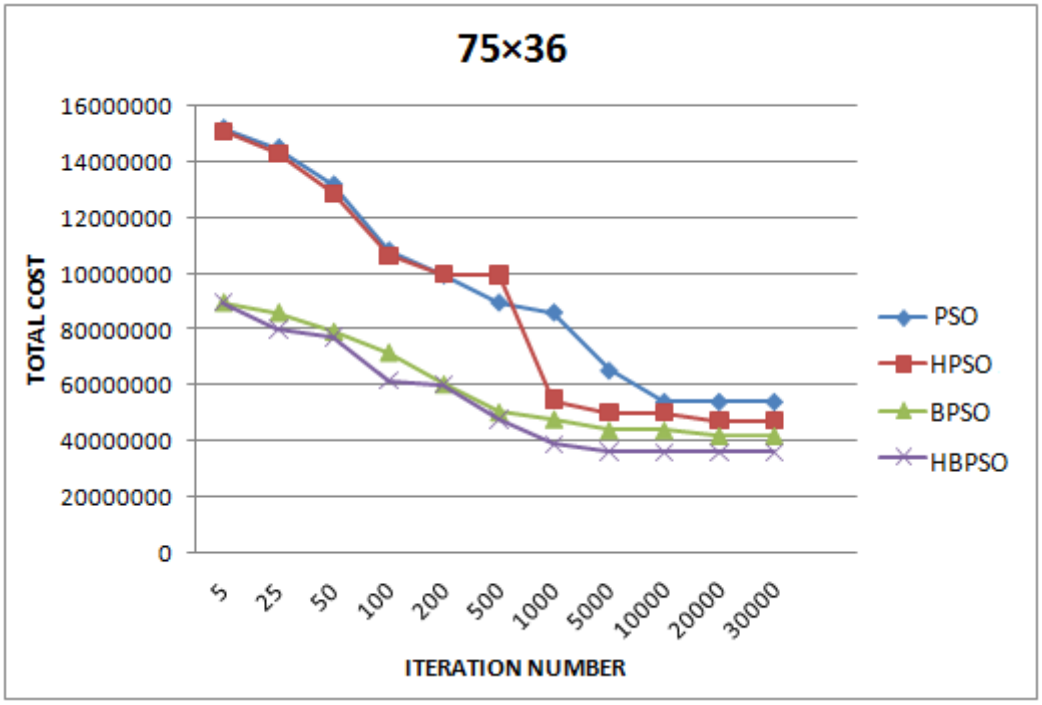


Figure 4.12 Convergence Graph of 75×36 problem

4.12. Comparison of Results

Table 4.18 shows the comparison of results among PSO, HPSO, BPSO and HBPSO and Table 4.19 shows the Percentage improvement in solution with HPSO, BPSO, and HBPSO when compared to PSO.

	BGA total cost	**HBGA total cost**	**BPSO total cost**	**HBPSO total cost**
7×6	9245	8320	8320	8320
50 × 12	204,140.90	191617.40	181,685.31	173753.29
39 × 12	197,410.34	177609.65	172,682.56	142889.60
75 × 36	54,344,516	47344516	41,817,140.0	36,205,080

Table 4.18. Comparison of results among PSO, HPSO, BPSO and HBPSO

	HPSO (% of improvement)	**BPSO (% of improvement)**	**HBPSO (% of improvement)**
7×6	10	10	10
50 × 12	6.13	11	16
39 × 12	10	12.5	28
75 × 36	12.8	23.06	33.38

Table 4.19 Percentage improvement in solution with HPSO, BPSO, and HBPSO compared to PSO.

4.13. Summary

An attempt is made successfully to apply PSO, HPSO, BPSO and HBPSO algorithms to address any kind of Lot Sizing problems which arise in a typical manufacturing industry. The proposed algorithm can be applied to any type of production system and any type of product structures. To the best of the knowledge of the author, such works have not been published so far in contemporary literature. This is the first work presenting an effective BPSO and HBPSO for Multi-level, Multi-item capacitated Lot sizing Problems.

CHAPTER 5

INTEGRATION OF INVENTORY AND SCHEDULING USING BPSO

5.1. Introduction

The inventory management literature is focused primarily on product. This literature aims to determine the quantity that should be delivered to a facility and the delivery timings. In inventory models, the manufacturing process of an item is treated as a black box which means the model is typically not concerned with the details of the manufacturing processes, and instead, focusses on upstream and downstream parts of the supply chain. The scheduling literature, on the contrary, is focused on the manufacturing process. In particular, it deals with the allocation of resources to the items to be manufactured and with the sequencing of production/ operation on each resource. Scheduling models can represent the availability of component parts using release dates, inventory costs of finished products using the notion of earliness and costs of delays in delivery of completed items using the concept of tardiness. However, they have traditionally not been concerned with the upstream and downstream parts of the supply chain unlike the inventory model. Thus, inventory management and scheduling have traditionally focused on different aspects of supply chain management and each of them have included only an abstract representation of the other in their respective models. In this work, we partially remove the barrier between the two fields by including Scheduling constraints in Inventory Optimization problem.

Lot sizing and scheduling is an elaborate process which deals with the determination of lots/ batches chosen for production/ manufacturing and assigning a sequence to them in order to meet the customer requirements in the most economical way. However, as simple as this sounds, the real job of lot sizing and scheduling is one of the most challenging task in the face of difficulties existing in the real world.

Few challenges on the same are being described; (i) Existence of high degree of uncertainties/ disruptions in supply chains e.g. uncertainties/ disruptions in supply chain between OEMs and prime contractors; between prime contractors and tier 1 suppliers and so on. (ii) Parts obsolescence, poor quality etc. which may throw sudden surprises and disrupt all the

assumptions made while taking lot sizing and scheduling decisions. (iii) multi-level BoM, multi-product manufacturing which react sharply to changes in external environment or factors which are beyond manufacturer's control. Therefore, it is essential that as many factors as possible should be considered while taking planning decisions. In other words, the decision process/ mechanism should be designed in such a way so that it integrates as many factors as possible during planning.

Production planning is first performed at a tactical level and, the different jobs are then supposed to be scheduled at the operational level. Therefore, the information about capacity requirements is available at tactical level and in an aggregate manner, this therefore does not guarantee that scheduling constraints are respected which may result in an infeasible production plan.

The decision variables, in inventory model, do not involve the decision on sequencing of operations within a period; therefore, an additional scheduling step is required to determine the same. The requirement of lot sizing and scheduling being done simultaneously is needed for cases in which set-up costs are dependent on chosen sequence. These types of problems are most commonly observed in process industries and have attracted the focus of recent researches on extending the CLSP to include decisions related to scheduling and sequence dependent setup costs. These problems are known as General Lot Sizing Problem or GLSP.

The production planning and scheduling in Material Requirement Planning (MRP) are performed separately in a manner as if these two activities are independent of each other. The output in such cases are typically a production plan which cannot be implemented (or is infeasible) since the scheduling constraints were not taken into account during planning. Therefore, a decision taken at tactical level fails to qualify for operationalization at operational level. This problem is further compounded by the fact that in MRP there is no constraint on capacity (capacity is treated as unlimited) when the decision on lot sizing is being taken. Therefore, once an output is obtained from MRP several changes are done to make the decision practical and feasible which results in unnecessary work and loss of productivity.

MRP utilizes several inputs (these inputs have been discussed in earlier chapters) to generate a production plan. There is also make-or-buy decision for some of the parts which affects the total cost and hence each of the inputs have to be precisely computed (for

inclusion/ exclusion) in order to have a good production plan. The importance of inventory and impact of its being excess/ low has long term implications on a company's profitability. In the light of this, significance of lot sizing and scheduling decisions, gain tremendous importance.

Scheduling is a critical process which ensures that resources are allocated to the production plan in the best economical way. Scheduling gives the information on when to initiate and terminate an operation/ process. Once the knowledge is gained on what to make and when, necessary materials, resources and auxiliary equipment can be readied, so that the production meets the planned deliveries.

However, one may encounter issues in terms of non-availability of material/ other resources which often results in disruption of the plan. As explained, this can be attributable to the fact that MRP takes no account of capacity constraints and operating sequences in its calculation. The sequencing of lots on the machine should be taken into account at the planning level (or when defining lot sizes) since there is a complex interaction between lot sizing and sequencing to achieve a good makespan. The earlier literatures prove that generating MRP and schedule at the same time can be a good alternative.

5.2. Implementation

From an operational point of view, production targets 'Pit' are utilized as inputs to scheduling and Pit is obtained by solving the production planning problem. This leads to a hierarchical information flow to scheduling from production planning. However, as already explained, the information pertaining to process capacity and cost has to be communicated to the production planning formulation (from scheduling) via the integration of some form of a scheduling model to get feasible and (near) optimal production targets. In principle, the two problems can be linked via constraints that enforce the production targets for each planning period are equal to the orders due at the end of the corresponding scheduling horizon.

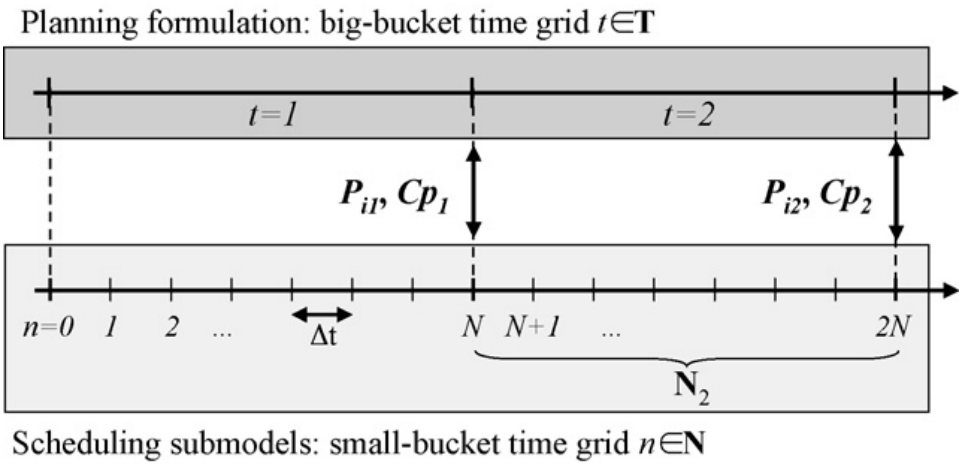


Figure 5.1 Integrated problems of planning and scheduling

It may be noted that in this, two time grids are utilized: a big-bucket planning grid (weeks or months) and a small-bucket scheduling grid (hours), with the former being the scheduling horizon of the latter. The linking constraints can be easily expressed, if a discrete-time scheduling grid is utilized. Considering the dynamic environment of market and production, constant updation of production targets should happen time-to-time. This is required to take care of the disturbances which arise in the form of delays in delivery of materials, failure of equipment etc. In this way, a production planning solution 'Pit' is typically used only for a few early periods, after which problem data is updated, and the problem is re-solved over a new planning horizon.

Integrating the planning model with detailed capacity constraints induced by the workshop topology and sequencing decisions: we consider necessary and sufficient conditions of feasibility of a production plan that theoretically gives permission to obtain global optimum solution to the production planning problem. Necessary conditions do not restrict sufficiently the set of feasible production plans. A production plan, solution to the simplified problem is not guaranteed to be achievable either. Sufficient conditions are written by fixing in advance a particular sequence of products on the machines. In doing so, exact detailed capacity constraints are easily expressed in the terms of the X (Quantity of items ordered) vector.

The difficulties of decisions at tactical level and operational level due to non-inclusion of capacity related inputs have already been explained.

An integrated approach which helps (to some extent) in bringing consistency between the two decisions (tactical and operational) of production management was presented. This

approach shall help in avoidance of the de-merits of the traditional approach where the tactical and operational decisions are taken in sequence. The objective here is to minimize the total cost of manufacturing which involves set up cost and carrying cost. Minimization of one cost affects the other cost.

5.3. Overview of the integration

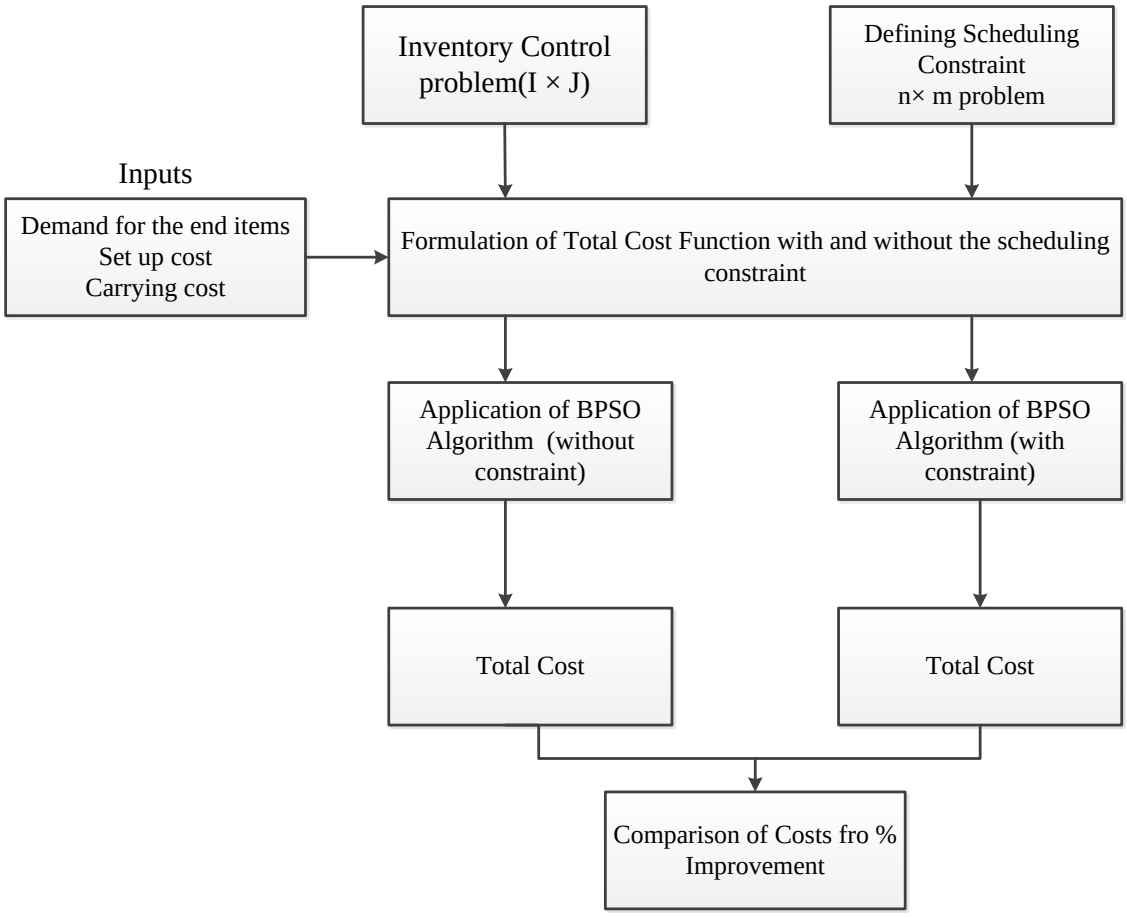


Figure 5.2 Overview of the integration.

BPSO technique is applied to different types and sizes of problems with and without scheduling constraint. The aim is to compare total cost of the plan with and without scheduling constraints. We tested for single item single level, multi-level and multi item problems by applying BPSO algorithm and the results have been compared for three different sizes of the problem.

5.4 Integration Formulation of Planning and Scheduling

To guarantee consistency between planning and scheduling decisions in complex production systems, detailed scheduling constraints are to be considered in the lot-sizing mathematical model (Lasserre 1992; Dauzère-Pérès and Lasserre 1994). They studied the impact of sequencing decisions in production planning. The basic idea is to observe that, for a fixed sequence of operation on the machines, the capacity constraints of the workshop are easily expressed in terms of the planning variables (quantities to produce at each period). They are just linear constraints in terms of these variables.

In the integrated Planning and scheduling model of a job shop, we must integrate the scheduling constraints in the planning problem. A set of sufficient condition is generated by imposing a fixed sequence of operations on the machines when a choice has to be made. In other words, before computing the production plan, we resolve the possible conflicts of operations on the machines.

The problem is formulated as

$$\min \sum_{t=1}^{T}\sum_{i=1}^{n}(c_{it}^{+}.I_{it}^{+} + c_{it}^{-}.I_{it}^{-} + c_{it}^{pr}.X_{it})$$

$$(I_{it}^{+} - I_{it}^{-}) - (I_{it-1}^{+} - I_{it-1}^{-}) - X_{it} + D_{it} = 0, i = 1,...n; t = 1,...,T \rightarrow (1)$$

$$X_{it} \geq 0, \forall i,t \rightarrow (2)$$

$$I_{it}^{+} \geq 0, \forall i,t \rightarrow (3)$$

$$I_{it}^{-} \geq 0, \forall i,t \rightarrow (4)$$

$$t_{ijkt} - t_{ij'k't} - p_{ij'k't}^{u}.X_{it} \geq 0, \forall (o_{ij'k't}, o_{ijkt}) \in A \rightarrow (5)$$

$$t_{ijkt} \geq 0, \forall o_{ijkt} \in N \rightarrow (6)$$

$$t_{ijkt} - t_{i'j'kt'} - p_{i'j'kt'}^{u}.X_{i't'} \geq 0, \forall (o_{i'j'kt'}, o_{ijkt}) \in S(y) \rightarrow (7)$$

$$t_{ijkt} + p_{ijkt}^{u}.X_{it} \leq \sum_{l=1}^{t} c_l, \forall o_{ijkt} \in L \rightarrow (8)$$

$$t_{ijkt} + p_{ijkt}^{u}.X_{it} \geq \sum_{l=1}^{t-1} c_l, \forall o_{ijkt} \in L \rightarrow (9)$$

The objective function in the above problem is the minimization of sum of the Inventory surplus, backlog, and production cost of the products to be planned. (1) is the standard inventory balance equation. Constraints 2, 3, 4 ensure that production items, inventory surplus, backlog quantities are always positive. Constraint 5 gives the conjunctive constraint

relationship among the operation on the machines. Constraint (6) ensures that starting times of operation Oijt are always positive. Constraint 7 gives the disjunctive constraints relations among the operations. Constraints 8 & 9 state that the last operations of the Jit must be completed in period t and not before. Constraint 7 replaced with necessary conditions which does not involve Disjunctive constraints.

$$\sum_{l=1}^{t}\Big(\sum_{o_{ijkl}\in O_{kl}} P_{ijkl}^{u}.X_{il}\Big) \le \sum_{l=1}^{t} C_l$$

The condition shown above ensures that sum of processing times of operations to be processed on the same machine cannot be larger than the time available. There are two solving procedures proposed by authors for the integrated model given above.

5.4.1 A One Pass Procedure for integrated problem

We know that, in solving Psc (y) for the fixed sequence y, the resulting plan P=(X, I) will be optimal with respect to this sequence. This plan is feasible since there exists at least one schedule (using the sequence y) which is compatible with that plan, i.e., all the jobs Ji associated with P are completed by their due date.

However, some demands may not be satisfied. The quality of the results strongly depends on the chosen sequence y. The procedure of one pass procedure is as follows:

- Solve Psc(y) with only constraints (1-4), i.e., ignoring the scheduling problem. The resulting plan P* is an ideal plan since no capacity constraint from the operational level is taken into account.
- Given this plan P*, which minimizes the make span. In general, P* is not feasible and some jobs will be completed after the due date.
- Solve Psc(y*) and take any optimal solution P1 as final production plan. This plan is now feasible since the constraints (9) are satisfied for the sequence y*.

As the sequence y* is optimal for the ideal plan p*, one may conjecture that y* is a good choice. The unfeasible ideal plan p* is adapted to the capacity constraints to yield a production plan P1.

5.5. Implementation of BPSO to Integrated Problems

PSO and BPSO algorithms have been discussed in the previous chapters. In the integration of Lot sizing and Scheduling, the representation of each particle is same and the execution is also similar except that scheduling constraints were included in the execution.

The following summarizes the PSO (BPSO procedure for integration).

Particle: is a candidate solution i in swarm at iteration k. The i^{th} particle of the swarm is represented by a d-dimensional vector and can be defined as $X_i^k = [X^k_{i1,}\ X^k_{i2,}\ X^k_{i3}\ldots.\ X^k_{id}]$, where x's are the optimized parameters and X^k_{id} is the position of the i^{th} particle with respect to d^{th} dimension. In other words, it is the value d^{th} optimized parameter in the i^{th} candidate solution.

Population: pop^k is the set of n particles in the swarm at iteration k, i.e., $pop^k = [\ X_1^k,\ X_2^k,\ X_3^k \ldots\ldots\ldots\ X_n^k]$.

Particle velocity: V_i^k is the velocity of particle i at iteration k. It can be described as $V_i^k = [V^k_{i1,}\ V^k_{i2,} V^k_{i3} \ldots V^k_{id}]$,where V_{id}^k is the velocity with respect to d^{th} dimension.

Particle best: PB_i^k is the best value of the particle i obtained until iteration k. The best position associated with the best fitness value of the particle i obtained so far is called particle best and defined as $PB_i^k = [pb^k_{i1,}\ pb^k_{i2,} pb^k_{i3} \ldots pb^k_{id}]$, with the fitness function $f(PB_i^k)$.

Global best: GB_i^k is the best position among all particles in the swarm, which is achieved so far and can be expressed as $GB_i^k = [gb^k_{i1,}\ gb^k_{i2,} gb^k_{i3} \ldots gb^k_{id}]$, with the fitness function $f(GB_i^k)$.

Termination criterion: it represents the condition in which the termination of search process will happen. In this study, the termination of search takes place when the iteration number reaches a maximum number which is a predetermined value.

The complete description of the binary PSO algorithm is given below:

Step 1: Initialization

a) Set n=twice the number of dimensions and k=0

b) 'n' particles are generated randomly as, $\{X_i^0, i=0,1,2,\ldots\ldots n\}$,where $X_i^0 = [X^0_{i1}, X^0_{i2}, X^0_{i3}\ldots. X^0_{id}]$.

c) The initial velocities are randomly generated for all particles, $\{V_i^0, i=0,1,2,\ldots\ldots n\}$, where $V_i^0 = [v^0_{i1}, v^0_{i2}, v^0_{i3}\ldots. v^0_{id}]$. v^0_{id} is randomly generated with $v_{id}=V_{min}+(V_{max} - V_{min})*rand()$.

d) The objective function, $f(X_i^0)$ will be Evaluated for each particle.

e) Set $PB_i^0 = X_i^0$,where $PB_i^0 = [pb^0_{i1}= X^0_{i1}, pb^0_{i2}= X^0_{i2}, pb^0_{i3}= X^0_{i3}\ldots.. pb^0_{id}= X^0_{id}$ along with its best fitness value, $f_i^{pbest}(PB_i^0, i=1,2,3\ldots.n)$.

f) Set the global best to , $f_i^{gbest}(GB^0)=\min\{ f_i^{pbest}(PB_i^0, i=1,2,3\ldots.n)\}$ with $GB^0=[gb_1, gb_{2,\ldots} gb_d]$

Step 2: Update iteration counter

- k=k+1

Step 3: Using the piece-wise linear function, update velocity.

$$\Delta v_{id}^k = c_1\, r_1\left(pb_{id}^{k-1} - X_{id}^{k-1}\right) + c_2\, r_2\left(gb_{id}^{k-1} - X_{id}^{k-1}\right) v_{id}^k = \quad \mathrm{h}\left(v_{id}^{k-1} + \Delta v_{id}^{k-1}\right)$$

- C_1, C_2 and r_1, r_2 are social cognitive parameters and uniform random numbers between respectively.

Step 4: Using the sigmoid function, update position.

- $X_{id}^k = \{1$, if $U(0,1) < \text{sigmoid}\ (v^k_{id})$
  0, otherwise

Step 5: Update particle best

- Each particle position and velocity are computed again with respect to its updated position and see if particle best will change. That is,
  If $f_i^k(X_i^k, i=0,1,2,\ldots\ldots n) < f_i^{pbest}(PB_i^{k-1}, i=0,1,2,\ldots\ldots n)$
  then
  $f_i^{pbest}(PB_i^k, i=0,1,2,\ldots\ldots n) = f_i^k(X_i^k, i=0,1,2,\ldots\ldots n)$
  else
  $f_i^{pbest}(PB_i^k, i=0,1,2,\ldots\ldots n) = f_i^{pbest}(PB_i^{k-1}, i=0,1,2,\ldots\ldots n)$

Step 6: Update global best

$f^{gbest}(GB^k) = \min\{ f_i^{pbest}(PB_i^k, i=1,2,\ldots\ldots n)\}$

if $f^{gbest}(GB^k) < f^{gbest}(GB^{k-1})$, then

$f^{gbest}(GB^k) = f^{gbest}(GB^k)$

else $f^{gbest}(GB^k) = f^{gbest}(GB^{k-1})$

Step 7: Stopping Criterion

- If the number of iteration exceeds the maximum number iteration, then stop, otherwise go to step 2.

5.5.1. BPSO Approach to the single level lot sizing problem

(a) Initial Solution Representation

Solution representation of particle p X^k_{id} is given in below table. This representation is due to Hernandez and Suer (1999), where each swarm contains 'P' number of particles referring to d dimensions and 'i' items. Here k represents iteration number.

A population of binary values (0 or 1) are randomly assigned for 'd' dimensions of 'i' items for Multi-level problems for all 'p' particles which gives the information about where setups are made.

If $R_{id}>0.5$ then $X_{id}=1$,else $X_{id}=0$

R_{id}= random value

i= iteration number= 1, 2, 3… n

k=iteration number=1, 2, 3… k

d=dimension of the particle (no of periods in the problem)=1, 2, 3…… t

For initial generation k=0, i.e., $X^k_{id}= X^0_{id}$

	1	2	3	4	……..	12
X^k_{0d}	1	0	0	1	……..	1
X^k_{2d}	-	-	-	-	-	-
X^k_{id}	-	-	-	-	-	-

According to particle solution, lot sizes are calculated as shown in below table. Time periods where demand is not 0 there set up has been made. So, order quantity in particular period may be (i) requirement of that period or (ii) requirement of that period including with group of requirements of periods ahead or (iii) zero.

Lot size according to particle dimension

	1	2	3	4	……..	12
Q^k_{1d}	140	0	0	170	……..	
Q^k_{2d}	-	-	-	-	-	-
Q^k_{3d}	-	-	-	-	-	-

Q^k_{id}=lot size of item I ordered in period d at iteration k of particle p.

(b) Velocity of initial generated particles

After assigning the particle dimensions, velocity values need to be calculated as shown in below table to find next generation population. This velocity calculation is of two types i.e., 1) velocity calculations for initial generation, 2) velocity calculation for remaining generations. Velocity values are restricted to some minimum and maximum as

V^k_{id} = [Vmin,Vmax] = [-5,5].

V^k_{id} = velocity of particle i in d_{th} dimension and at iteration k.

For initial generation velocity values are calculated using following formula

$V_{id}=V_{min}+ (V_{max} - V_{min})*R$

R= a random value within 0 to 1, which is generated using rand ().

Velocity matrix of particle

	1	2	3	4	……..	12
V^k_{1d}	-2.1	1.6	2.8	-3.6	……..	1.85
V^k_{2d}	-	-	-	-	-	-
V^k_{3d}	-	-	-	-	-	-

(c) Finding Particle Best and Global Best

Particle having best fitness value [$f(X^k_p)$] is assigned to global best. As it is the initial generation, all particle best values are equal to particle values as shown in the below table.

	1	2	3	4	……..	12
PB^k_{1d}	1	0	0	1	……..	1
PB^k_{2d}	-	-	-	-	-	-

Among the obtained particle best values, we find the minimum value which is global best value of that population.

	1	2	3	4	……..	12
GB^k_{1d}	1	0	0	1	……..	1

(d) Updating velocity and position for next generations

1. Updating velocity (V^k_{id}):

Velocity values from previous generation to new generation are updated using Piece wise linear function

$$\Delta v_{id}^k = c_1\ r_1\left(pb_{id}^{k-1} - X_{id}^{k-1}\right) + c_2\ r_2\left(gb_{id}^{k-1} - X_{id}^{k-1}\right) v_{id}^k = \quad \mathrm{h}\left(v_{id}^{k-1} + \Delta v_{id}^{k-1}\right)$$

C_1 and C_2 are social and cognitive parameters and r_1 and r_2 uniform random numbers between (0, 1).

$h(V_{id}^k)$= Vmax if V_{id}^k > Vmax

= V_{id}^k if $|V_{id}^k| \leq$ Vmax

= Vmin if V_{id}^k < Vmin.

2. Updating Position (X^k_{id}) by sigmoid function

X^k_{id}=1, if R< $S(V_{id}^k)$

= 0 otherwise

Sigmoid Function S (V_{id}^k):

This function forces velocity values to be in the limits of '0' to '1'. It helps to update generation of X^k_{id} values.

Sigmoid (V_{id}^k) = $\frac{1}{1+e^{-V_{id}^k}}$ $\frac{1}{1+e^{-V_{id}^k}}$

(e) Updating particle best and Global Best (PB^k_{id}, GB^k_{id}):

After each and every iteration, update particle best and global best values according to the fitness values of particles in the newly generated swarm.

(f) Termination:

If the number of iterations reached a predetermined value, called maximum number of iterations then stop searching, otherwise go to (d).

5.5.2. Numerical Example

A problem with d=5 periods, c=1, A=100 has been constructed to demonstrate the working of the binary PSO algorithm in solving the LSP (lot sizing problem). The complete computational flow for illustration purpose is given below:

Step 1:

- k=0, and n=3
- The initial particles in the swarm generated randomly are X_1^0, X_2^0, $X_3^0$and corresponding velocities V_1^0, V_2^0, V_3^0which are given as follows:

	d	**1**	**2**	**3**	**4**	**5**
X_1^0	x_{1d}	1	1	0	0	0
V_1^0	v_{1d}	2.90	1.60	-1.80	3.5	-1.60
X_2^0	x_{2d}	1	0	1	0	0
V_2^0	v_{2d}	3.80	2.90	3.00	-0.70	-1.20
X_3^0	x_{3d}	1	0	0	1	1
V_3^0	v_{3d}	-3.0	1.50	-2.70	2.00	1.40

- Computation of the fitness function for the particles are given in below:

d	**1**	**2**	**3**	**4**	**5**	**6**	$f(X_i^k)$
R_d	100	60	40	50	80	70	
x_{id}^k	1	0	1	0	1	0	
v_{id}^k	3.8	2.9	3.0	-0.7	-1.2	3.1	
Q_{id}^k	160		90		150		
I_{id}^k	60		50		70		
cI_{id}^k	60		50		70		
Ax_{id}^k	100		100		100		
$C\left(X_i^k\right)$	160		150		170		480

- For each solution in the population we calculate the fitness value known as particle best

	d	**1**	**2**	**3**	**4**	**5**	$f_i^{pbest}(PB_i^0)$
PB_1^0	pb_{1d}	1	1	0	0	0	580
PB_2^0	pb_{2d}	1	0	1	0	0	470
PB_3^0	pb_{3d}	1	0	0	1	1	440

- Among the above fitness values, we find min fitness value as the Global best

	d	**1**	**2**	**3**	**4**	**5**	$f^{gbest}(GB^0)$
GB^0	gb_d	1	0	0	1	1	440

Step 2: Updating iteration, k=k+1

Step 3: Updating velocity values using piece wise linear function.

- Assuming C_1, C_2, r_1, $r_2 = 0.5$
- By using the piece wise linear function, we update the velocity value for next generation

$$\Delta v_{12}^0 = 0.5 * 0.5\left(pb_{12}^0 - x_{12}^0\right) + 0.5 * 0.5\left(gb_2^0 - x_{12}^0\right)$$

$$\Delta v_{12}^0 = 0.5 * 0.5(1-1) + 0.5 * 0.5(0-1) = -0.25$$

$$v_{12}^1 = h\left(v_{12}^0 + \Delta v_{12}^0\right) = h(1.6 + (0 - 0.25)) = 1.35$$

Step 4: Updating position using sigmoid function

$$Since\ U(0,1) = 0.99 < sigmoid\left(v_{12}^1 = 1.35\right) = 0.79$$

- After updating all velocity and position values the next generation values are noted.

Step 5: update particle best

	d	**1**	**2**	**3**	**4**	**5**	$f_i^{pbest}(PB_i^1)$
PB_1^1	pb_{1d}	1	0	0	1	0	420
PB_2^1	pb_{2d}	1	0	1	1	0	440
PB_3^1	pb_{3d}	1	0	0	1	1	440

Step 6: Update Global Best

	d	**1**	**2**	**3**	**4**	**5**	$f^{gbest}(GB^0)$
GB^0	gb_d	1	0	0	1	0	420

Step 7: iterations up to termination criterion.

5.5.3. BPSO Implementation for Multi-Level Problem

In this example, two items are there; item 2 is having independent demand and demand of the other item depends on the first one. The BOM Structure and Demand values are as shown below.

	1	2	3	4	5	6
Demand	20	100	50	30	10	100

Item	S.C	H.C
1	500	50
2	100	10

Product Demand, set up and Holding Cost

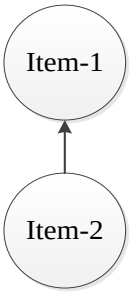


Figure 5.3 BOM Structure for 2×6 problem

Step 1: Initial Generation

Particle 1

Item-1	1	0	0	0	1	1
	200	0	0	0	10	100
	2.9	-1.8	3.5	-1.2	0.7	3.8
Item-2	1	0	0	0	1	0

	200	0	0	0	110	0
	-1.5	1.6	-2.7	3.3	1.8	-3.9

Fitness function = $f(X_1^0)$ = 17200

Particle 2

Item-1	1	1	1	0	0	1
	20	100	90	0	0	100
	-2.5	1.7	3	-4	1.6	2.3
Item-2	1	0	0	0	0	1
	210	0	0	0	0	100
	2.2	-3.9	1.4	3	-4.6	2

Fitness function = $f(X_2^0)$ = 7500

Particle 3

Item-1	1	0	1	0	1	1
	120	0	80	0	10	100
	3	-4.6	2	-1.2	0.7	3.8
Item-2	1	0	1	0	1	0
	120	0	80	0	110	0
	-3	-4	1.6	2.9	-1.8	3.5

Fitness function = $f(X_3^0)$ = 9800

Step 2: As it is initial generation, assign each particle in the swarm tp particle best (PB)

$pb^{1,0}_{1,1} = X^{1,0}_{1,1}$,$pb^{1,0}_{1,2} = X^{1,0}_{1,2}$, …….. $pb^{1,0}_{1,12} = X^{1,0}_{1,12}$

$pb^{1,0}_{2,1} = X^{1,0}_{2,1}$,$pb^{1,0}_{2,2} = X^{1,0}_{2,2}$, …….. $pb^{1,0}_{2,12} = X^{1,0}_{2,12}$

	d	1	2	3	4	5	6	Fitness
$pb^{1,0}_{id}$	i=1	1	0	0	0	1	1	17200

	i=2	1	0	0	0	1	0	
$pb^{2,0}_{id}$	i=1	1	1	1	0	0	1	7500
	i=2	1	0	0	0	0	1	
$pb^{3,0}_{id}$	i=1	1	0	1	0	1	1	9800
	i=2	1	0	1	0	1	0	

	d	1	2	3	4	5	6	Fitness
$GB^{2,0}_{id}$	i=1	1	1	1	0	0	1	7500
	i=2	1	0	0	0	0	1	

Step 3: Update velocity using Piece wise linear function.

- Assuming C_1, C_2, r_1, $r_2 = 0.5$

Update particle velocity

$$\Delta v^{0}_{12} = 0.5 * 0.5\left(pb^{0}_{12} - x^{0}_{12}\right) + 0.5 * 0.5\left(gb^{0}_{2} - x^{0}_{12}\right)$$

$\Delta V^{1,0}_{12}$= 1*0.5(0-0)+1*0.5(1-0)=0.5

$V^{1,1}_{12}=P(V^{1,0}_{12}+\Delta V^{1,0}_{12})$ = P(-1.8+0.5)=-1.3

Update particle Position

R(0,1)=0.11<Sigmoid (-1.3)=0.21

So new position value = $X^{1}_{1,2}$=1

After calculating velocity and position values of all dimensions, updated particles are

	D	1	2	3	4	5	6	Fitness
$X^{1,1}_{id}$	i=1	1	1	1	1	0	1	6400
	$V^{1,1}_{1d}$	2.9	-1.3	4	-1.2	0.2	3.8	
	Sig $V^{1,1}_{1d}$	0.94	0.21	0.9	0.23	0.54	0.97	
	Random	0.97	0.11	0.4	0.2	0.67	0.8	
	i=2	1	0	0	0	0	1	
	$V^{1,1}_{2d}$	-1.5	1.6	-2.7	3.3	1.3	3.3	
	Sig $V^{1,1}_{2d}$	0.18	0.83	0.06	0.96	0.78	0.96	
	Random	0.1	0.91	0.1	0.99	0.8	0.91	
$X^{2,1}_{id}$	i=1	1	1	1	1	0	1	3500
	i=2	1	1	1	1	0	1	

$X^{3,1}_{id}$	i=1	1	0	0	0	1	1	9800
	i=2	1	0	0	0	1	0	

Updated particle best matrix

	d	1	2	3	4	5	6	Fitness
$pb^{1,1}_{id}$	i=1	1	1	1	1	0	1	6400
	i=2	1	0	0	0	0	1	
$pb^{2,1}_{id}$	i=1	1	1	1	1	0	1	3500
	i=2	1	1	1	1	0	1	
$pb^{3,1}_{id}$	i=1	1	0	1	0	1	1	9800
	i=2	1	0	1	0	1	0	

Update the global best value

	d	1	2	3	4	5	6	Fitness
$GB^{2,1}_{id}$	i=1	1	1	1	1	0	1	3500
	i=2	1	1	1	1	0	1	

Step 4: Termination

Repeat this procedure (step 3) until iteration number k< maximum iteration.

Integrated problem are solved by using the exact capacity constraint from a standard scheduling problem to the above lot sizing problem. In the planning problem, amount ordered at every period should consider the makespan of that product on the machine shop. That makespan is taken as a constraint in planning problem. Thereby planning problem respects the exact capacity constraint given in scheduling.

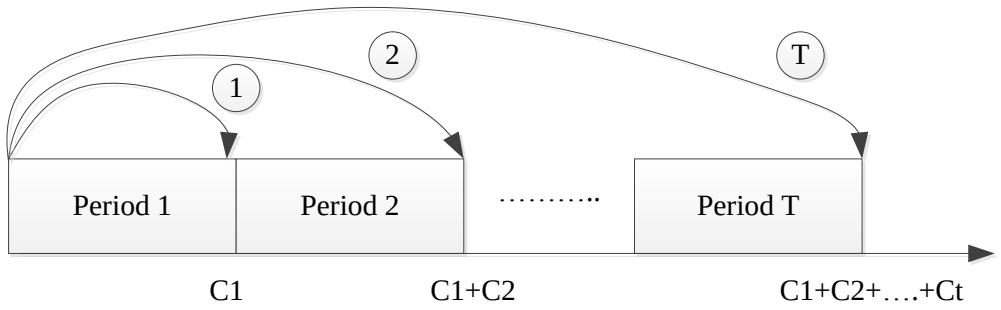


Figure 5.4 Representation of planning problem with constraints at every period

5.6. Results and Analysis

Planning and scheduling consistency decisions in complex production systems are guaranteed when detailed scheduling constraints are to be considered in lot-sizing mathematical models. We have considered three types of problems in lot sizing and included exact capacity constraint from a Benchmark scheduling problem. We have tested three different product structures in each type of problem without any scheduling constraint and compared with the results we obtained by including the scheduling constraints. 7×6 problem is taken from jinxing xie, jiefang dongs heuristic Algorithm for general lot sizing problem(2002), remaining problems are considered randomly.

All the problems are solved using Binary Particle Swarm Optimization. The algorithm is coded in C language and run on intel® Core™ i3 processor 2.20 GHz with 4GB RAM.

5.6.1 Single item single level Problem

Single-level, single-item problems are having only one product to be planned at each period. Here Table 5.1 shows the Demand of end products and costs involved in three different problems and Table 5.2 shows the comparison of results for the same problem.

Period	1	2	3	4	5	6	7	8	9	10	11	12
Demand (1×12)1	15	5	55	110	65	165	125	25	90	15	140	115
Demand (1×12)2	25	20	15	10	20	35	30	38	45	65	30	25
Demand (1×12)3	50	50	50	50	50	50	50	50	50	50	50	50

Table 5.1 Demand of products and cost involved in (1x12) problem

Problem	Setup cost	Holding Cost
Demand (1×12)1	S.C=780	H.C=97.83
Demand (1×12)2	S.C=500	H.C=82.35
Demand (1×12)3	S.C=385	H.C=68.59

Table 5.2 Comparison of results

Comparison of results with and without scheduling constraint tested at different iterations

iteration No	Simple (1×12)1	Integrated (1×12)1	Simple (1×12)2	Integrated (1×12)2	Simple (1×12)C	Integrated (1×12)C
5	19830.44	19726.94	7558.75	7495.02	7664.5	7531
25	13163.21	12954.31	6454.78	6248.35	6353.24	6219.37
50	10047.45	10033.95	5758.22	5678.25	5763.42	5536.74
100	9069.15	9064.65	5326.32	5240.35	5269.15	5118.21
500	9068.76	9063.28	5176.64	4976.21	4620	4500
1000	9068.76	9062.85	5174.51	4954.82	4620	4500

Table 5.3 (1×12) problem solution with three different problems

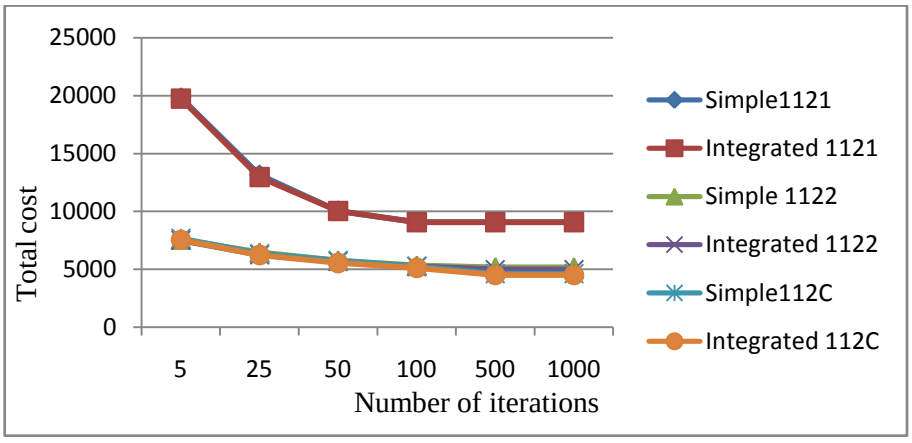


Figure 5.5 Convergence of three (1×12) problem solutions at different iterations

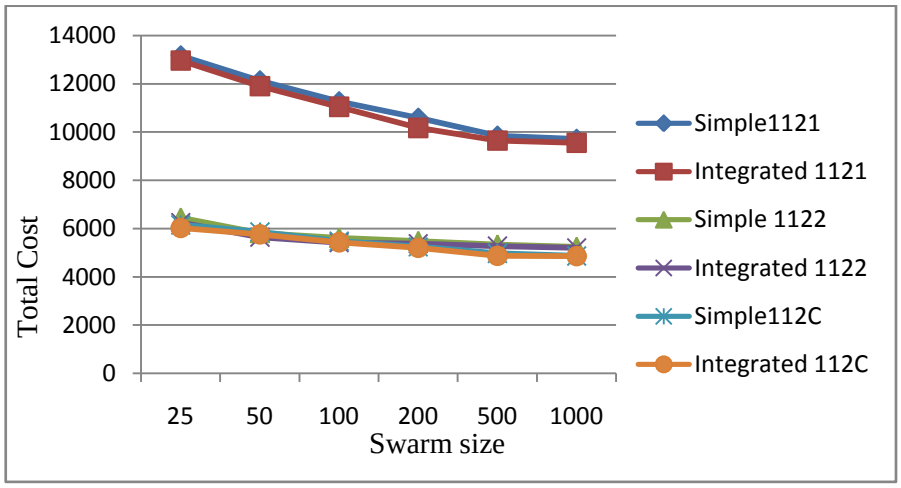


Figure 5.6 Comparison of three (1×12) problem solutions at different Swarm sizes

Observation:

- The total cost is observed to be reducing when number of iterations are increasing.
- No improvement in solution is observed when number of iterations are beyond 500.
- Total cost is decreasing when swarm size is increasing. It is observed that the optimum swarm size for this problem where there is no further improvement in solution.

5.6.2. Single-item, Multi-level Problem

These types of problems involve one end product and having more number of products involved in getting end product. The end product is having independent demand and lower level products are having dependent demands. Figure 5.7 and 5.8 show the BOM structure of 7×6 and 40×12 problems, Table 5.4 shows the demand for end products and cost involved in 7×6 problems. Table 5.5 shows the end product demand, costs for three different sizes of problems.

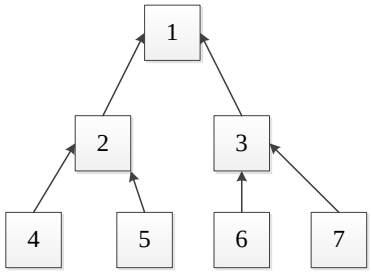


Figure 5.7 BOM Structure of 7×6 problem

Period	1	2	3	4	5	6	
Problem76	25	20	15	10	20	35	
Item no	1	2	3	4	5	6	7
S.C	400	500	1000	300	200	400	100
H.C	12	0.6	1	0.04	0.03	0.04	0.04

Table 5.4 Demand of products and cost involved in (7×6) problem

period	1	2	3	4	5	6	7	8	9	10	11	12
Demand 25×12	15	5	15	110	65	165	125	25	90	15	140	115
Demand 40×12	10	100	10	130	115	150	70	10	65	70	165	25
Demand 50×12	15	5	15	120	65	155	125	25	95	15	135	115

Table 5.5 Demand of products for three different problems

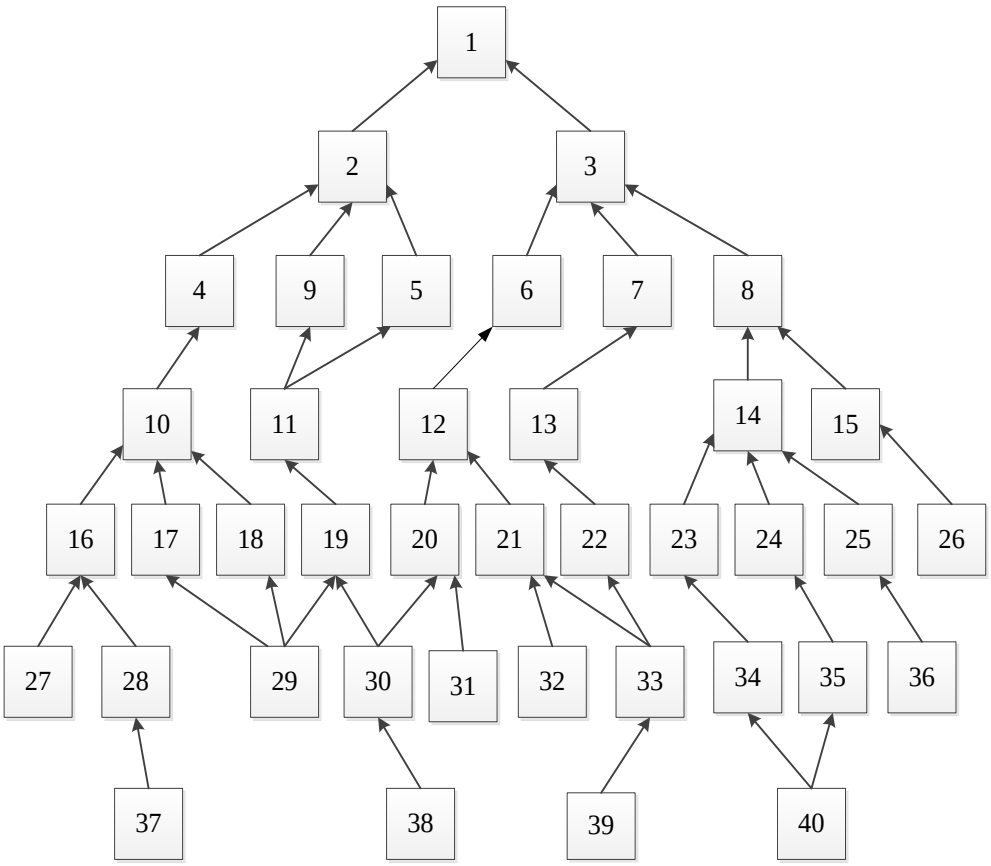


Figure 5.8 BOM Structure of 40×12 problem

Item No	Problem 50×12		Problem 40×12		Problem 25×12	
	S.C	H.C	S.C	H.C	S.C	H.C
1	780	97	410	40.08	780	97.83

2	200	45	450	35.27	200	45.19
3	590	43	90	59.66	590	43.2
4	710	5	140	25.42	710	5.82
5	890	26	880	10.42	890	26.04
6	610	18	440	22.64	610	18.87
7	920	27	70	22.31	920	27.03
8	210	15	430	19.53	210	15.64
9	490	2	930	1.34	490	2.67
10	920	1	650	25.12	920	1.86
11	520	23	740	9.46	520	23.5
12	540	12	680	17.48	540	12.59
13	50	25	800	4.32	510	25.13
14	500	16	220	14.28	500	16.42
15	300	1	850	2.56	300	0.84
16	450	1.5	400	10.07	450	1.02
17	440	0.5	650	4.59	440	0.62
18	510	23	860	7.13	510	23.71
19	910	15	850	8.82	910	15.32
20	830	20	670	10.61	830	20.58
21	730	8	370	6.02	730	8.71
22	80	3	360	2.78	850	3.14
23	450	1	310	2.95	450	0.94
24	30	13	440	9.32	370	13.02
25	390	7	590	0.31	390	7.34
26	540	7	580	1.45		
27	160	4	650	3.63		
28	40	18	450	4.35		
29	410	5	820	3.29		
30	140	2	620	5.04		
31	390	6	580	2.53		
32	370	15	850	3.3		
33	520	4	340	0.61		

34	700	3	80	2.52
35	160	6	690	4.83
36	20	3	430	3.44
37	420	2	60	0.91
38	160	2	760	2.64
39	450	6	180	2.65
40	340	7	150	2.5
41	750	3		
42	140	1		
43	430	3		
44	890	2		
45	850	2		
46	240	1		
47	890	0.5		
48	610	4		
49	860	4		
50	350	2		

Table 5.6 Set up and Holding cost involved in three SIML problems

Comparison of results without and with scheduling constraint tested at number of iterations

iteration No	simple 50×12	Integrated 50×12	simple 40×12	Integrated 40×12	simple 25×12	Integrated 25×12	Simple 7×6	Integrated 7×6
5	256564.83	240840	367011.844	366131.8438	306744.94	299342.5	5235.25	5190
25	244064.84	220425	333017.781	332167.7813	260425.02	229615	4323.5	4285.27
50	221292.95	214097.5	276022.781	274832.7813	217487.42	187560	3500	3465.15
100	208054.88	208940	251291.906	250011.9063	193371.08	166397.5	2965.32	2846.58
500	204272.	197120	233231.1	231861.18	166823.	158520	2833.	2742.88

	45		88	75	33		63	
1000	200755.48	196262.5	221354.688	220344.6875	162807.89	152022.5	2795.34	2731.85
2000	194767.62	193652	218257.954	217965.7056	159652	150712.24		
5000	193546.38	192758	214125.36	213924.1637	159242.85	150215.32		

Table 5.7 SIML problem solution with four different sizes at different iterations

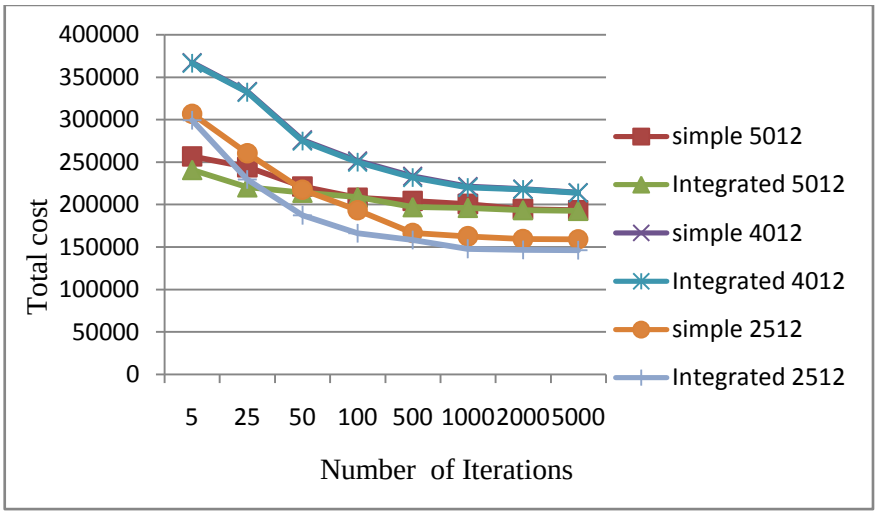


Figure 5.9 Four SIML problem-solutions at number of iterations and their comparisons.

Effect of Swarm Sizes

Comparison of results with and without scheduling constraint tested at different Swarm sizes

swarm size	Simple 50×12	Integrated 50×12	simple 40×12	Integrated 40×12	simple 25×12	Integrated 25×12	simple 7×6	Integrated 7×6
25	204272.45	197120	232350.0156	225665.9688	193371.08	187560.32	5235.25	5190
50	197210.38	190990	223641.25	219219.37	186374.25	181753.51	4821.35	4768.51
100	192461.23	184746	218421.812	215734.88	182369.75	178257.45	4587.68	4412.76
200	185741.36	180638	214723.56	211845.25	180684.521	176852.47	4458.97	4332.57
500	183876.	178965	208675.6	206196.4	178963.3	175245.	4407.	4296.48

	24		9	8	5	68	27	
1000	181637.52	177642	201634.32	200861.79	177821.24	175021.82		
5000	181471.69	177361.78	201438.21	200524.98	177605.21	173954.76		

Table 5.8 SIML problem solution with four different sizes at different swarm sizes

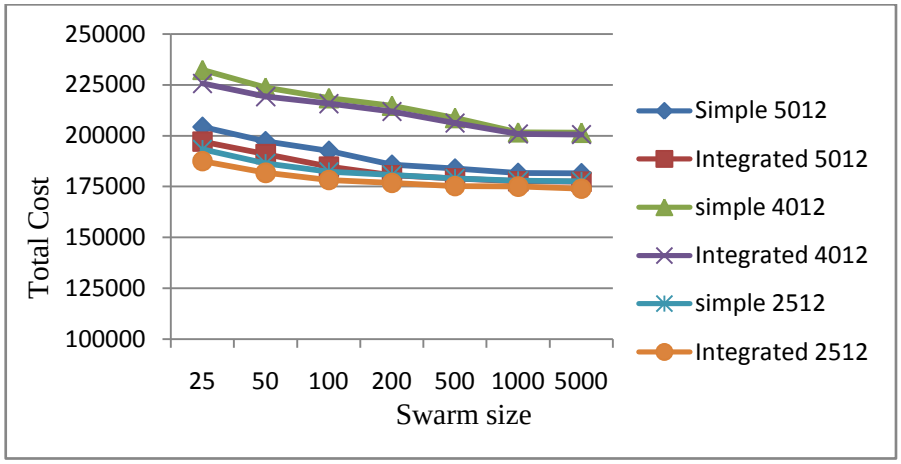


Figure 5.10 Convergence of 3- SIML problems solutions at various Swarm sizes

Observation:

- The total cost is observed to be reducing when number of iterations are increasing.
- No improvement in solution is observed when number of iterations are beyond 2000.
- Total cost is decreasing when swarm size is increasing. It is observed that the optimum swarm size for this problem where there is no further improvement in solution is 1000.

5.6.3. Multi item level Problem

These types of problems involve one or more number of end products and more levels of products connected with end products. The end product is having independent demand and lower level products are having dependent demands. Here Figure 5.8 shows the BOM structure for 39×12 problems. Table 5.7 and 5.8 shows the end product demand, Set up cost and Holding Cost for three different sizes of problems.

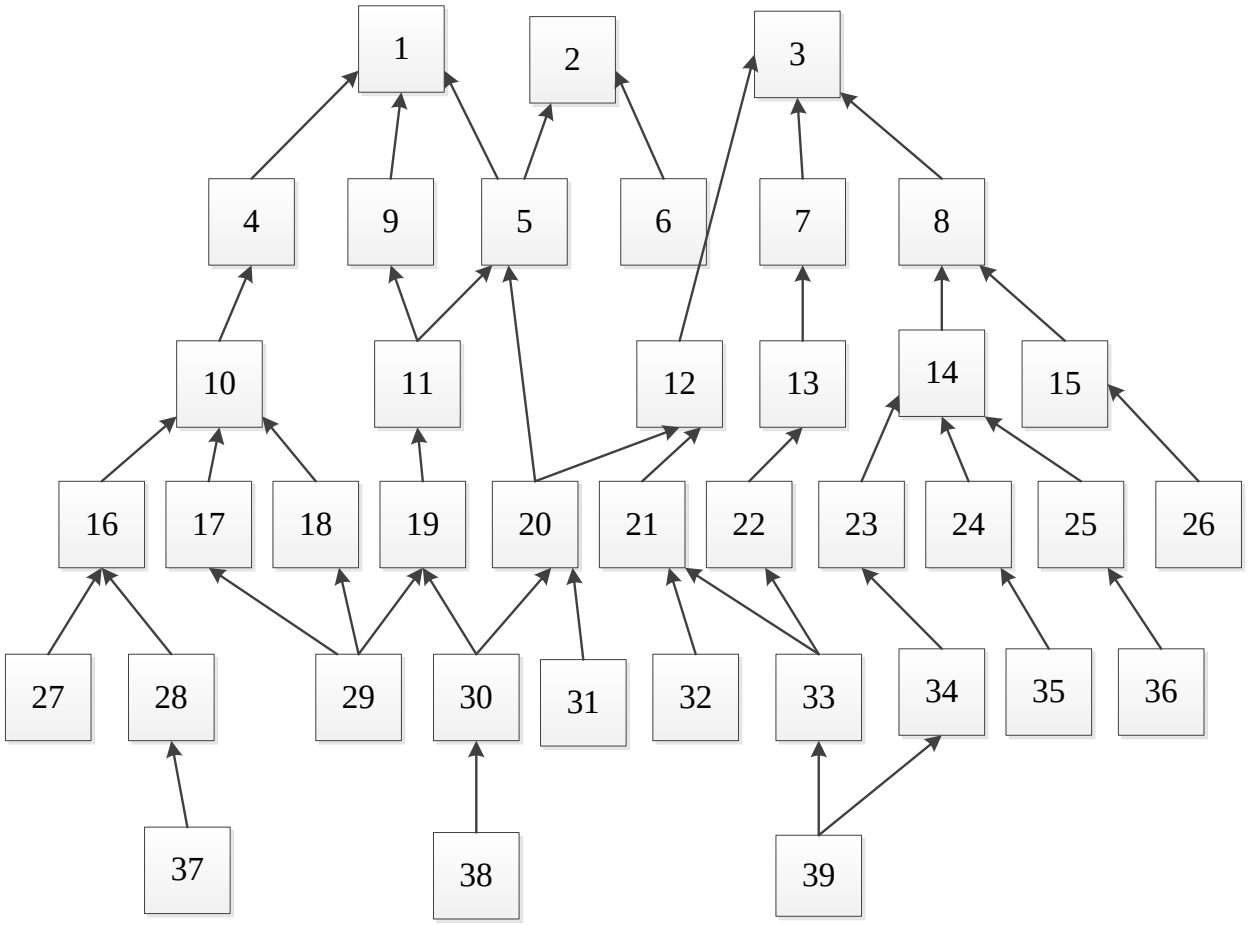


Figure 5.11 BOM structure of 39×12 problem

Period	1	2	3	4	5	6	7	8	9	10	11	12
Item 1	10	100	10	130	115	150	70	10	65	70	165	25
Item 2	175	15	85	90	85	90	75	150	75	10	150	15
Item 3	135	165	15	105	25	120	50	60	5	140	60	10

Table 5.9 Demand of products involved in 39×12 problems

Item No	Problem 39×12		Problem 25×12		Problem 15×12	
	S.C	H.C	S.C	H.C	S.C	H.C
1	410	40.08	410	40.08	410	40.08
2	450	35.27	450	35.27	450	35.27

3	90	59.66	90	59.66	90	59.66
4	140	25.42	140	25.42	140	25.42
5	880	10.42	880	10.42	880	10.42
6	440	22.64	440	22.64	440	22.64
7	70	22.31	70	22.31	70	22.31
8	430	19.53	430	19.53	430	19.53
9	930	1.34	930	1.34	930	1.34
10	650	25.12	650	25.12	650	25.12
11	740	9.46	740	9.46	740	9.46
12	680	17.48	680	17.48	680	17.48
13	800	4.32	800	4.32	800	4.32
14	220	14.28	220	14.28	220	14.28
15	850	2.56	850	2.56	850	2.56
16	400	10.07	400	10.07		
17	650	4.59	650	4.59		
18	860	7.13	860	7.13		
19	850	8.82	850	8.82		
20	670	10.61	670	10.61		
21	370	6.02	370	6.02		
22	360	2.78	360	2.78		
23	310	2.95	310	2.95		
24	440	9.32	440	9.32		
25	590	0.31	590	0.31		
26	580	1.45				
27	650	3.63				
28	450	4.35				
29	820	3.29				
30	620	5.04				
31	580	2.53				
32	850	3.3				
33	340	0.61				
34	80	2.52				

35	690	4.83
36	430	3.44
37	60	0.91
38	760	2.64
39	180	2.65

Table 5.10 Costs of products involved in three different problems sizes

iteration no	Simple 39×12	Integrated 39×12	Simple 15×12	Integrated 15×12	Simple 25×12	Integrated 25×12
5	422388.1563	412894.2812	267876.5625	252053.1563	350988.8125	336092.2813
25	376876.25	365135.0937	204949.6563	194649.6563	293958.2188	283658.2188
50	340013.4688	328273.875	163244.7031	156844.7031	236183.5313	222883.5313
100	280418.8438	267891.5937	124069.1641	123669.1641	194280.3594	189580.3594
200	255216.375	244539.421	121145.0547	120225.8984	181030.5625	175430.5625
500	244271.6406	231291.8906	113352.3047	112148.213	170112.3906	166256.7813
1000	230098.312	223192.4062	103500.2031	101742.2361	156994.7188	154257.6875
2000	222510	213361.4687	100472.1862	94232.80469	155365.7031	150367.7969

Table 5.11 MIML problem solution with three different sizes at different iterations

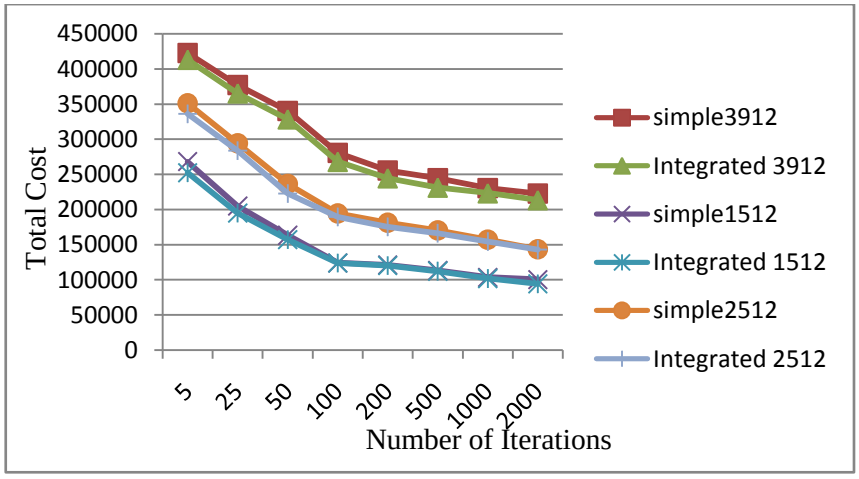


Figure 5.12 Convergence of 3- MIML problem – solutions convergence at different iterations

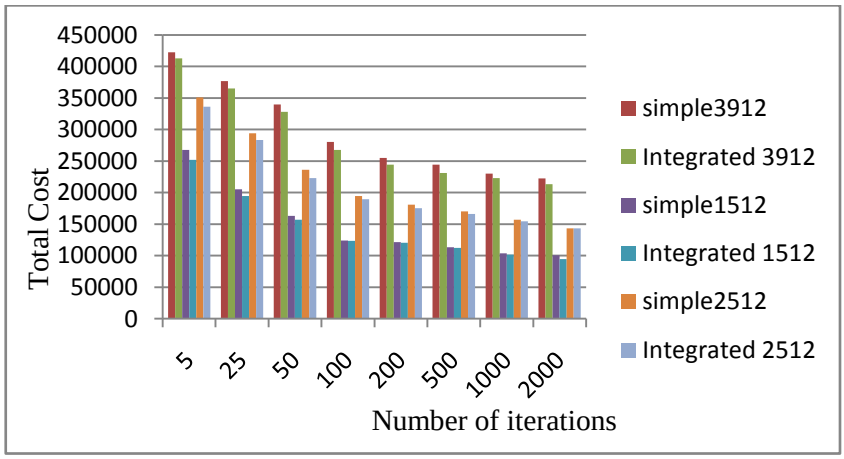


Figure 5.13 Three MIML problems - solutions at number of iterations and their comparison.

Observation

- The total cost is observed to be reducing when number of iterations are increasing.
- No improvement in solution is observed when number of iterations are beyond 2000.

Effect of Swarm Sizes

Comparison of results without and with scheduling constraint tested at various Swarm sizes

Swarm Size	Simple 39×12	Integrated 39×12	Simple 25×12	Integrated 25×12	Simple 15×12	Integrated 15×12
25	279391.5625	267891.5937	194280.3594	189580.3594	164144.7031	155214.7031
50	267341.4062	254709.4219	182030.5625	176430.5625	134069.1641	133529.1641
100	261467.37	249491.508	172132.3906	167276.7813	122175.0547	121245.8984
200	258461.0706	244861.9716	157914.7188	155257.6875	119352.3047	118148.213
500	244539.421	236291.8906	145365.7031	144067.7969	116500.2031	115742.2361
1000	235483.217	228424.1875	143154.28	142075.84	114072.1862	112232.8047
5000	234615.78	227531.54	142549.71	141287.61	113186.2146	111682.73

Table 5.12 MIML problem solution with three different sizes at different swarm sizes

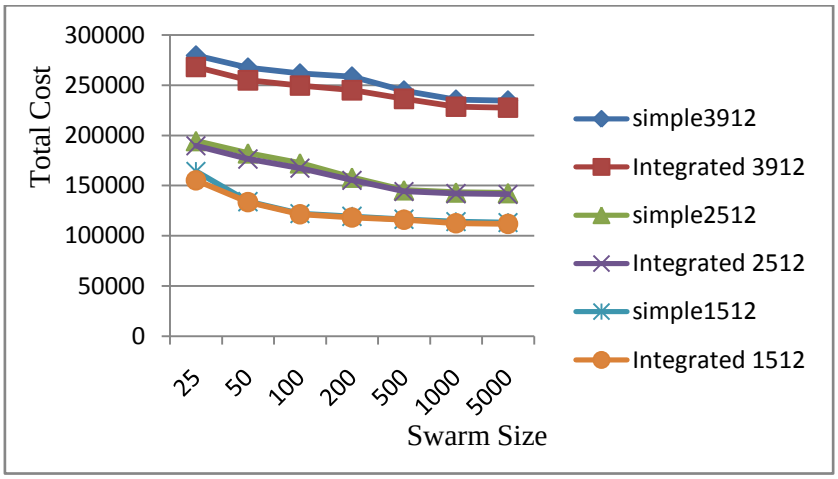


Figure 5.14 Convergence of 3 MIML problems - solutions at different Swarm sizes

Observation:

- The total cost is observed to be reducing when Swarm size is increasing.
- It is observed that the optimum swarm size for this problem where there is no further improvement in solution is 1000.

5.6.4. Comparison of Results

We have tested Single level, Multi-level single item and multi item problems with product structures by assuming demand and cost for each problem. Each problem is tested with and without introducing the scheduling constraint. Table 5.13 shows the percentage improvement of solutions when the problems are tested with scheduling constraint at their Global best values.

SINGLE LEVEL SINGLE ITEM			
Problem	Simple	Integrated	% of Improvement
112-1	9068.76	9062.85	0.06
112-2	5174.51	4954.82	4.24
112-C	4620	4500	2.59
SINGLE LEVEL MULTI ITEM			
Problem	Simple	Integrated	% of Improvement
50×12	193546.4	192758	0.40
40×12	218258	213924.16	1.98

25×12	159242.9	150215.32	5.66
7×6	2795.34	2731.85	2.27
MULTI LEVEL MULTI ITEM			
Problem	Simple	Integrated	% of Improvement
39×12	222510	213361.47	4.11
25×12	100472.2	94232.805	6.21
15×12	155365.7	150367.8	3.21

Table 5.13 percentage improvement in solutions for three types of problems

5.7. Summary

BPSO/HBPSO technique has been implemented successfully for integration of different L.P problems like single item multi-level (SIML) and single item single level (SISL), multi item multi-level (MIML) problems with three product structures with a scheduling constraint. We found reduction in inventory cost by considering the scheduling constraint. We have tested a very few problem instances. However, we have to test thoroughly a large number of scheduling constraints in lot sizing problem for effective evaluation of integration of scheduling and lot sizing. We found that convergence of solution takes place at large number of iterations and size of swarm. The author emphasizes that there is no work reported on the integration problem using BPSO/HBPSO technique in the contemporary literature.

CHAPTER 6

CONCLUSIONS AND SCOPE FOR FUTURE WORK

6.1. Conclusions

In this work, the author has used particle swarm optimization and binary particle swarm algorithms. The hybrids of these algorithms namely HPSO and HBPSO were developed. All the algorithms were applied on:

i. Job shop scheduling problems
ii. Lot sizing problems
iii. Integrated lot sizing and scheduling problems.

The codes were developed in MATLAB and run on intel core 2 DeoT6400@2.0 GHz computer. Each problem was made to run 30 times on each algorithm to reduce redundancy. Some of the important conclusions drawn are given below:

1. 250 bench mark problems of JSSP available in the literature were thoroughly tested using PSO, HPSO, BPSO and HBPSO algorithms. The problems were tested with different sizes of initial population. After thorough testing, we found that the population size of initial pod is best equal to number of operations of n x m problem for fast convergence to an optimal solution. In other situations, the chances of getting local optimal solutions are more. Simulated annealing and iterative improvement methods were combined in the execution of PSO and BPSO algorithms and thus HPSO and HBPSO algorithms were developed.

2. 250 bench mark instances of JSSPs were tested using PSO, HPSO, BPSO, HBPSO and the consolidated result is given below:

Total problems tested = 250 Bench Mark Instances			
Method	**No. of solutions equal to BKS**	**No. of solutions not equal to BKS**	**% of Confirmation**
PSO	143	137	57.2

HPSO	246	4	98.4
BPSO	212	38	84.8
HBPSO	237	13	94.8

It shows that HPSO and HBPSO are very good algorithms as they were more than 94.15% (≈95%) confirming with Best Known Solution (BKS) values. There is no algorithm existing which produces results which are 100% equal to BKS. The proposed algorithms have proved 95% efficiency in generating BKS for 250 Bench Mark Instances. Hence, it is proved that HPSO and HBPSO are excellent to address Job Shop Scheduling problems.

3. Ten tough problems given by Yamada, 1997 i.e. ABZ7, ABZ8, ABZ9, LA21, LA24, LA25, LA29, LA38 and LA40 were also solved by PSO, HPSO, BPSO and HBPSO and it was observed that BPSO, HPSO and HBPSO algorithms produce BKS values for these tough problems. Hence, the algorithms developed are robust and able to solve any kind of JSSPs effectively.

4. PSO was hybridized with simulated annealing and the resultant algorithm is called Hybridized PSO (HPSO). HPSO shows an improvement of 42% compared to PSO. Similarly, HBPSO is produced by applying iterative improvement in BPSO. HBPSO produces 12% improvement over BPSO. We also found that HBPSO and HPSO are fast converging as compared to BPSO and PSO. Hence, they emerge as time and solution efficient algorithms. These algorithms are recommended to solve any kind of Job shop scheduling problems.

5. PSO, HPSO, BPSO and HBPSO techniques have been successfully employed to model and simulate all sorts of LS problems like single-item, multi-level (SIML) and single-item, single-level (SISL), multi-item, multi-level (MIML), uncapacitated and capacitated problems under consideration to minimize total cost. BPSO, HBPSO techniques give faster convergence when compared to PSO and HPSO techniques.

6. Among the solutions obtained for the all problems under consideration in lot sizing by PSO, HPSO, BPSO and HBPSO methods, it has been observed that though all four techniques are successful methods in obtaining solutions, the solution obtained by HBPSO is unique and more efficient in solving small, medium and large size Lot sizing

problems. Thus, HBPSO proves to be a robust solution method for Lot sizing problems in general. It is also found that BPSO is taking minimum CPU time for solving small and medium Lot Sizing Problems. However, the solution efficiency is same with HBPSO.

7. Computational experience shows that HPSO and HBPSO algorithms can be implemented as separate optimization modules for solving all types of JSSP and Lot Sizing Problems independently in SAP/MRP-II packages.

8. An attempt is successfully made to implement Binary Particle Swarm Optimization (BPSO) algorithm and its variant HBPSO to address the integration of lot sizing and scheduling problems that arise in a typical manufacturing industry. Scheduling decisions are respected during the decision taken for lot sizing problems by taking a capacity constraint similar to the known scheduling problem. We have integrated the problem up to some extent which leads to the reduction of cost involved in Lot sizing problems. The proposed algorithm can be applied to any type of production system and product structure.

9. BPSO/HBPSO technique has been successfully implemented for integration of different LS problems like single-item, multi-level (SIML) and single-item, single-level (SISL), multi-item, multi-level (MIML) problems with three product structures with a scheduling constraint. We found reduction in inventory cost by considering the scheduling constraint. We have tested a very few problem instances. However, we have to test thoroughly a large number of scheduling constraints in lot sizing problem for effective evaluation of integration of scheduling and lot sizing. We found that convergence of solution takes place at large number of iterations and sizes of swarm. The author emphasizes that there is no work reported on the integration problem using BPSO/HBPSO technique in the contemporary literature.

6.2. Scope for future work

Other efficient soft computing algorithms like Invasive Weed Optimization, Music Based Harmonic Search and symbiosis algorithm etc. and their invariants may be applied to JSSP and lot sizing problems independently and jointly to test their performances on complex problems.

Considering the JSSP and lot sizing problems as multiobjective in nature, parato optimal set of solutions can be obtained using PSO, HPSO, BPSO and HBPSO and other new algorithms. The methods developed need to be tested on real time real world problems of industry where lot of constraints and random demands need to be satisfied.

REFERENCES

1. A. C. Coello, D. C. Rivera, and N. C. Cortes, "Use of an artificial immune system for job shop scheduling," in Proc. 2nd Int. Conf. Artificial Immune Syst., 2003, vol. 2787, pp. 1–10.
2. A. Floudas, Deterministic Global Optimization: Theory, Methods and Applications, vol. 37 of Nonconvex Optimization and Its Applications, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000.
3. A.S Jain, S.Meeran Mascis, Dario Pacciarelli "Deterministic Job-shop scheduling : Past , Present and Future" European Journal of Operational Research, Volume 113, 1999, Pages 390-434
4. Adil Baykasoğlu, Lale Özbakır, Türkay Dereli "Multiple Dispatching Rule based heuristic for Multiobjective Scheduling of Job Shops using Tabu Search" 5th International Conference on Managing Innovations in Manufacturing, September 2002
5. Afentakis. P. and Gavish. B, (1986). Optimal lot-sizing algorithms for complex product structures. Operation Research, 34(2):237–249.
6. Ajith; G. Crina; R. Vitorino (éditeurs), Stigmergic Optimization, Studies in Computational Intelligence, volume 31, 299 pages, 2006. ISBN 978-3-540-34689-0
7. Alain Hertz, Marino Widmer "An improved tabu search approach for solving the job shop scheduling problem with tooling constraints" Discrete Applied Mathematics, Volume 65, Issues 1-3, 7 March 1996, Pages 319-345.
8. Alessandro Mascis, Dario Pacciarelli "Job-shop scheduling with blocking and no-wait constraints" European Journal of Operational Research, Volume 143, Issue 3, 16 December 2002, Pages 498-517.
9. Ali Allahverdi, C.T. Ng, T.C.E. Cheng, Mikhail Y. Kovalyov "A survey of scheduling problems with setup times or costs" European Journal of Operational Research, Volume 187, Issue 3, 16 June 2008, Pages 985-1032.
10. Anthony Caumond, Philippe Lacomme, Nikolay Tchernev, A memetic algorithm for the job-shop with time-lags Computers & Operations Research, Volume 35, Issue 7, July 2008, Pages 2331-2356.
11. B.Karimi, S.M.T. Fatemi Ghomi, J.M.Wilson," The capacitated lot sizing problem: a review of models and algorithms", the international journal of Management science, Omega 31(2003) 365-378.
12. Bagheri, M. Zandieh, I. Mahdavi, M. Yazdani, An artificial immune algorithm for the flexible jobshop scheduling problem Future Generation Computer Systems, In Press, Accepted Manuscript, Available online 15 October 2009.
13. Bernard Roy "The out-ranking approach and the foundation of ELECTRE methods" In Bana e Costa ,C.A.,(ed.) reading in Multiple Criteria Decision Aid, 1990

14. Betul Yagmahan, Mehmet Mutlu Yenisey "A multi-objective ant colony system algorithm for flow shop scheduling problem" Expert Systems with Applications, In Press, Corrected Proof, Available online 11 July 2009
15. Bi-objective localization: D. Manjarres, J. Del Ser, S. Gil-Lopez, M. Vecchio, I. Landa-Torres, S. Salcedo-Sanz, R. Lopez-Valcarce, "On the Design of a Novel Two-Objective Harmony Search Approach for Distance- and Connectivity-based Node Localization in Wireless Sensor Networks", Engineering Applications of Artificial Intelligence, in press, June 2012.
16. Blazewickz J (1996) ,"The job shop scheduling problem: Conventional and new solutions techniques", Eur J Oper Res pp 931–30.
17. Bostjan Murovec, Peter suhel "A repairing technique for the local search of the job-shop problem" European Journal of Operational Research, Volume 153, Issue 1, 16 February 2004, Pages 220-238.
18. Bruker P (1995), "Scheduling algorithms 2nd edn. Springer", Berlin Heidelberg New York
19. C. Gicquel, Capacitated lot sizing models: a literature review, hal-00255830, version 1 - 14 Feb 2008.
20. Carlier J, Pison E (1989) ,"An algorithm for solving the job shop problem", Manage Sci 35:164–176
21. Castro L. de and Timmis J., 2003. "Artificial Immune Systems as a Novel Soft Computing Paradigm". Soft Computing Journal, vol. 7, Issue 7.
22. Ching-Fang Liaw "hybrid genetic algorithm for the open shop scheduling problem" European Journal of Operational Research, Volume 124, Issue 1, 1 July 2000, Pages 28-42.
23. Christian Almeder, A Hybrid Optimization Approach for Multi-Level Capacitated Lot-Sizing Problems, European Journal of Operational Research, 2009,599-606
24. Christos T. Maravelias , Charles Sung, Integration of production planning and scheduling: Overview, challenges and opportunities, Computers and Chemical Engineering 33 (2009) 1919–1930.
25. Coleman, B.J., McKnew, M.A., 1991. An improved heuristic for multilevel lot sizing in material requirements planning. Decision Sciences 22, 136–156.
26. D. Tarantilis, C. T. Kiranoudis "A list-based threshold accepting method for job shop scheduling problems" International Journal of Production Economics, Volume 77, Issue 2, 21 May 2002, Pages 159-171.
27. D.Cartysse, J.Maes, L.V.Van Wassenhove, Set partitioning and column generation heuristic for capacitated dynamic Lot Sizing, European Journal of Operations Research 46(1990)38-48.

28. D.Y. Sha, Cheng-Yu Hsu, A hybrid particle swarm optimization for job shop scheduling problem Computers & Industrial Engineering, Volume 51, Issue 4, December 2006, Pages 791-808.

29. Dam Scheduling: Geem, Z. W. "Optimal Scheduling of Multiple Dam System Using Harmony Search Algorithm", Lecture Notes in Computer Science, 2007.

30. Dasgupta D, Forrest S. "Artificial immune systems in industrial applications". Proc Second International Conference on Intelligent Processing and Manufacturing Materials (IPMM), Honolulu, July 1999, pp10–15

31. Dasgupta, D. 'An Overview of Artificial Immune Systems and Their Applications', In Artificial Immune Systems and Their Applications, D. Dasgupta (ed.), Springer-Verlag, 1999, pp. 3 – 21.

32. Dasgupta, D" Special Issue on Artificial Immune System" IEEE Trans. Evol. Comput (IEEE Transactions on Evolutionary Computation), 6, 3 (2002), 225-256.

33. Dauzère- Pérès, S. and Lasserre, J.B.: On the importance of sequencing decisions in production planning and scheduling, International Transaction in Operational Research Vol. 9, No. 6 (2002) 779-793.

34. Davide Anghinolfi, Massimo Paolucci, A new discrete particle swarm optimization approach for the single-machine total weighted tardiness scheduling problem with sequence-dependent setup times European Journal of Operational Research, Volume 193, Issue 1, 16 February 2009, Pages 73-85.

35. De Castro. L., and Timmis, J. (2002) 'Artificial Immune Systems: A New Computational Intelligence Approach' by Springer-Verlag, London, September, 35 p.

36. De Castro. L.N., Von Zuben, F.J "Learning and Optimization Using the Clonal Selection Principle". IEEE Trans. Evol. Comput (IEEE Transactions Evolutionary Computation), 6, 3 (2002), 239-251.

37. Design of radar codes: S. Gil-Lopez, J. Del Ser, S. Salcedo-Sanz, A. M. Perez-Bellido, J. M. Cabero and J. A. Portilla-Figueras, "A Hybrid Harmony Search Algorithm for the Spread Spectrum Radar Polyphase Codes Design Problem", Expert Systems with Applications, Volume 39, Issue 12, pp. 11089–-11093, September 2012.

38. Dirk C. Mattfeld, Christian Bierwirth "An efficient genetic algorithm for job shop scheduling with tardiness objectives" European Journal of Operational Research, Volume 155, Issue 3, 16 June 2004, Pages 616-630.

39. Dongyun Wang, Liping Liu, Hybrid particle swarm optimization for solving resource-constrained FMS Progress in Natural Science, Volume 18, Issue 9, 10 September 2008, Pages 1179-1183.

40. Drexl A & Kimms A, Lot sizing and scheduling - Survey and extensions, European Journal of Operational Research 99, pp 221-235, 1997.

41. Edwin D. Gomez urrutia, Riad Aggoune Stephane Dauzere-Peres, New integrated approach for solving multi-level lot sizing and scheduling problems, 9th International Conference of Modeling, Optimization and Simulation - MOSIM'12, June 6-8, 2012.

42. Edwin David Gómez Urrutiaa, Riad Aggounea and Stéphane Dauzère-Pérèsb, Solving the integrated lot-sizing and job-shop scheduling problem, International Journal of Production Research, 2014.

43. Efficient design of open Wifi networks: I. Landa-Torres, S. Gil-Lopez, J. Del Ser, S. Salcedo-Sanz, D. Manjarres, J. A. Portilla-Figueras, "Efficient Citywide Planning of Open WiFi Access Networks using Novel Grouping Harmony Search Heuristics", accepted for its publication in Engineering Applications of Artificial Intelligence, May 2012.

44. El-Bouri, N. Azizi, S. Zolfaghari, A comparative study of a new heuristic based on adaptive memory programming and simulated annealing: The case of job shop scheduling European Journal of Operational Research, Volume 177, Issue 3, 16 March 2007, Pages 1894-1910.

45. Erschler JF, Roubellat JP, Vernhes (1976),"Finding some essential characteristics of the feasible solutions for a scheduling problem", Operations Research 24:774–783

46. Face milling: Zarei, O., Fesanghary, M., Farshi, B., JaliliSaffar, R. and Razfar, M.R. "Optimization of multi-pass face-milling via harmony search algorithm", Journal of Materials Processing Technology, In press.

47. Ferdinando Pezzella, Emanuela Merelli "A tabu search method guided by shifting bottleneck for the job shop scheduling problem" European Journal of Operational Research, Volume 120, Issue 2, 16 January 2000, Pages 297-310.

48. Fleischmann B & Meyr H, The General Lot sizing and Scheduling Problem, OR Spektrum 19,Vol. 1, pp. 11-21, 1997.

49. Frank Werner, Andreas Winkler "Insertion techniques for the heuristic solution of the job shop problem" Discrete Applied Mathematics, Volume 58, Issue 2, 24 March 1995, Pages 191-211.

50. French S (1982),"Sequencing and scheduling: An introduction to the mathematics of the job shop", Wiley, New York

51. Garey M, et al (1976), "The complexity of flow shop and job shop scheduling", Math Oper Res 1:117–129

52. Gary I. Green, Leonard B. Appel, An alternative framework to Lagrangian relaxation approach for job shop scheduling European Journal of Operational Research, Volume 149, Issue 3, 16 September 2003, Pages 499-512.

53. Ground Water Modeling: Ayvaz, M. T. "Simultaneous Determination of Aquifer Parameters and Zone Structures with Fuzzy C-Means Clustering and Meta-Heuristic Harmony Search Algorithm", Advances in Water Resources, 2007.

54. Gu nter Fandel, Cathrin Stammen-Hegene, Simultaneous lot sizing and scheduling for multi-product multi-level production, Int. J. Production Economics 104 (2006) 308–316.

55. Guohui Zhang, Xinyu Shao, Peigen Li, Liang Gao An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem Computers & Industrial Engineering, Volume 56, Issue 4, May 2009, Pages 1309-1318.

56. H. Sarper, M. C. Henry N. Jawahar, P. Aravindan, S.G. Ponnabalm, "A genetic algorithm for scheduling manufacturing system", International Journal of advanced Manufacturing Technology,1998,vol.14,588- 607.

57. Haoxun Chen, Peter B. Luh, An efficient dynamic dispatching rule for scheduling in a job shop International Journal of Production Economics, Volume 32, Issue 3, November 1993, Pages 301-313.

58. Heinz Gröflin, Andreas Klinkert "Feasible insertions in job shop scheduling, short cycles and stable sets" European Journal of Operational Research, Volume 177, issue 2, 1 March 2007, Pages 763-785.

59. Heinz Gröflin, Andreas Klinkert, Nguyen Pham Dinh "Feasible job insertions in the multi-processor task job shop" European Journal of Operational Research, Volume 185, Issue 3, 16 March 2008, Pages 1308-1318.

60. Héla Ouerfelli, Abdelaziz Dammak, Emna Kallel Chtourou / Benders-based approach for an integrated Lot-Sizing and Scheduling problem. © International Journal of Combinatorial Optimization Problems and Informatics, Vol. 3, No. 3, Sep-Dec 2012.

61. Helena Ramalhinho Lourenço "Job-shop scheduling: Computational study of local search and large-step optimization methods" European Journal of Operational Research, Volume 83, Issue 2, 8 June 1995, Pages 347-364.

62. Hernández, W. and G. Süer, "Genetic Algorithms in Lot Sizing Decisions", Proceedings of the Congress on Evolutionary Computing (CEC99), Washington DC, July 6-9, 1999.

63. Hong Zhou, Waiman Cheung, Lawrence C. Leung, Minimizing weighted tardiness of job-shop scheduling using a hybrid genetic algorithm European Journal of Operational Research, Volume 194, Issue 3, 1 May 2009, Pages 637-649.

64. Hossein Hajimirsadeghi, "A Hybrid PSO/IWO algorithm for fast and global optimization" in proc. EUROCON 2009

65. Ikou KAKU, Zhaoshi LI and Chunhui XU, Solving Large Multilevel Lot-sizing Problems with an Effective Heuristic Algorithm Based on Segmentation, Innovative Computing Information and Control (ICICIC'08) 978-0-7695-3161-8/08 $25.00 © 2008 IEEE

66. J. C. Spall, Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control, Wiley-Interscience Series in Discrete Mathematics and Optimization, Wiley-Interscience, Hoboken, NJ, USA, 2003.

67. J. H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to B, control, and Artificial Intelligence, University of Michigan Press, Ann Arbor, Mich, USA, 1975.

68. Jacek Blazewicz, Erwin Pesch, Malgorzata Sterna "The disjunctive graph machine representation of the job shop scheduling problem", European Journal of Operational Research, Volume 127, Issue 2, 1 December 2000, Pages 317-331.

69. Jalil Layegh, Fariborz Jolai, Mohsen Sadegh Amalnik, A memetic algorithm for minimizing the total weighted completion time on a single machine under step-deterioration Advances in Engineering Software, Volume 40, Issue 10, October 2009, Pages 1074-1077.

70. Jason Chao-Hsien Pan, Han-Chiang Huang "A hybrid genetic algorithm for no-wait job shop scheduling problems" Expert Systems with Applications, Volume 36, Issue 3, Part 2, April 2009, Pages 5800-5806.

71. Jean-Paul Watson, J. Christopher Beck, Adele E. Howe, L. Darrell Whitley "Problem difficulty for tabu search in job-shop scheduling"Artificial Intelligence, Volume 143, Issue 2, February 2003, Pages 189-217.

72. Jie Gao, Mitsuo Gen, Linyan Sun, Xiaohui Zhao "A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems" Computers & Industrial Engineering, Volume 53, Issue 1, August 2007, Pages 149-162.

73. Jin-hui Yang, Liang Sun, Heow Pueh Lee, Yun Qian, Yan-chun Liang, Clonal Selection Based Memetic Algorithm for Job Shop Scheduling Problems, Journal of Bionic Engineering, Volume 5, Issue 2, June 2008, Pages 111-119.

74. Jinxing Xie And Jiefang Dong, Heuristic Genetic Algorithm for General Capacitated Lot Sizing problem, Computers and Mathematics with Applications 44(2002) 263-276.

75. Johann Hurink, Sigrid Knust "Tabu search algorithms for job-shop problems with a single transport robot" European Journal of Operational Research, Volume 162, Issue 1, 1 April 2005, Pages 99-111.

76. John B. Jensen, Efficient dispatching rules for scheduling in a job shopInternational Journal of Production Economics, Volume 48, Issue 1, 10 January 1997, Pages 87-105.

77. Jörg Homberger, Decentralized multi-level uncapacitated lot-sizing by automated negotiation, 4OR-Q J Oper Res (2010) 8:155–180

78. Jorge M.S. Valente, Rui A.F.S. Alves, Filtered and recovering beam search algorithms for the early/tardy scheduling problem with no idle time Computers & Industrial Engineering, Volume 48, Issue 2, March 2005, Pages 363-375.

79. Joseph Begnaud, Saif Benjaafar, The multi-level lot sizing problem with flexible production Sequences, IIE Transactions (2009) 41, 702–715

80. Ju-Seog Song, Tae-Eog Lee "A tabu search procedure for periodic job shop scheduling" Computers & Industrial Engineering, Volume 30, Issue 3, July 1996, Pages 433-447.

81. K. Lee and Z. Geem, "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice", Computer Methods in Applied Mechanics and Engineering 194, pp.3902-3933,2005.

82. K. M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," IEEE Control Systems Magazine, vol. 22, no. 3, pp. 52–67, 2002.

83. K. Steinhöfel, A. Albrecht, C. K. Wong "Two simulated annealing-based heuristics for the job shop scheduling problem"European Journal of Operational Research, Volume 118, Issue 3, 1 November 1999, Pages 524-548.

84. Kendall G. and Soubeiga E. and Cowling P.. "Choice function and random hyperheuristics". 4th Asia-Pacific Conference on Simulated Evolution and Learning SEAL 2002: 667--671.

85. Kennedy, J., Eberhart, R., and Shi, Y. (2006). Swarm intelligence. Handbook of Nature-Inspired and Innovative Computing, 187–219.

86. Khald Mesghouni, Pierre Borne "Evolutionary Algorith For Job shop scheduling" Int.j.appl.Math.Comput.Sci.vol.14,2004,Pages 91-103

87. Kimms A, A genetic algorithm for multi-level, multi-machine lot sizing and scheduling, Computers & Operations Research 26, pp. 829-848, 1999.

88. Kirkpatrick, S.; C. D. Gelatt, M. P. Vecchi (1983-05-13). "Optimization by Simulated Annealing". Science. New Series 220 (4598): 671-680. Retrieved 2009-01-16.

89. Klorklear Wajanawichakon, Rapeepan Pitakaso, Solving large unconstrainted multi level lot-sizing problem by a binary particle swarm optimization. International Journal of Management Science and Engineering Management, 6(2): 134-141, 2011.

90. Krasnogor N. (1999). "Coevolution of genes and memes in memetic algorithms ocean". Graduate Student Workshop: 371.

91. Kun FAN, Ren-qian ZHANG, Guo-ping XIA Solving a Class of Job-Shop Scheduling Problem based on Improved BPSO Algorithm Systems Engineering - Theory & Practice, Volume 27, Issue 11, November 2007, Pages 111-117.

92. Lars Mönch, René Drießel "A distributed shifting bottleneck heuristic for complex job shops" Computers & Industrial Engineering, Volume 49, Issue 3, November 2005, Pages 363-380.

93. Lars Mönch, Rene Schabacker, Detlef Pabst, John W. Fowler "Genetic algorithm-based sub problem solution procedures for a modified shifting bottleneck heuristic for complex job shops" European Journal of Operational Research, Volume 177, Issue 3, 16 March 2007, Pages 2100-2118.

94. Lotfi Gaafar, Applying genetic algorithms to dynamic lot sizing with batch ordering, Computers & Industrial Engineering 51 (2006) 433–444

95. M. Chandrasekaran. P. Asokan . S. Kumanan . T. Balamurugan . S. Nickolas "Solving job shop scheduling problems using artificial immune system" 8 February 2005 / Accepted: 30 July 2005 / Published online: 3 January 2006 # Springer-Verlag London Limited 2006.

96. M. Dorigo et L.M. Gambardella, Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem, IEEE Transactions on Evolutionary Computation, volume 1, numéro 1, pages 53-66, 1997.

97. M. Dorigo, G. Di Caro, and L. M. Gambardella, "Ant algorithms for discrete optimization," Artificial Life, vol. 5, no. 2, pp. 137–172, 1999.

98. M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," IEEE Transactions on Systems, Man, and Cybernetics. Part B, vol. 26, no. 1, pp. 29–41, 1996.

99. M. Fatih Taşgetiren & Yun-Chia Liang, a binary particle swarm optimization algorithm for the lot sizing problem, Journal of Economic and Social Research 5 (2), 1-20.

100. M. Ghirardi, C. N. Potts, Makespan minimization for scheduling unrelated parallel machines: A recovering beam search approach European Journal of Operational Research, Volume 165, Issue 2, 1 September 2005, Pages 457-467.

101. M. Rabbani, M. Aramoon Bajestani, G. Baharian Khoshkhou, A multi-objective particle swarm optimization for project selection problem Expert Systems with Applications, Volume 37, Issue 1, January 2010, Pages 315-321.

102. M. Zlochin, M. Birattari, N. Meuleau, et M. Dorigo, Model-based search for combinatorial optimization: A critical survey, Annals of Operations Research, vol. 131, pp. 373-395, 2004.

103. M. Gopalakrishnan, K.Ding, J.M.Bourjolly, S.Mohan, A tabu search heuristic for the capacitated Lot Sizing problem with set up carryover, Management Science 47(6) (2001)851-863

104. M. Mahdavi, M.Fesanghary, E.Damangir, "An improved harmony search algorithm for solving optimization problems", Applied Mathematics and Computation188, pp.1567–1579,2007.

105. Makoto Asano, Hiroshi Ohta, A heuristic for job shop scheduling to minimize total weighted tardiness Computers & Industrial Engineering, Volume 42, Issues 2-4, 11 April 2002, Pages 137-147.

106. Martens, M. De Backer, R. Haesen, J. Vanthienen, M. Snoeck, B. Baesens, Classification with Ant Colony Optimization, IEEE Transactions on Evolutionary Computation, volume 11, number 5, pages 651—665, 2007.

107. Meyr H, Simultaneous Lot sizing and Scheduling by Combining Local Search with Dual Reoptimization, European Journal of Operational Research 120, pp 311-326, 2000.

108. Mohammad Mohammadi, Integrating lotsizing, loading, and scheduling decisions in flexible flow shops, International Journal of Advanced Manufacturing Technology (2010) 50:1165–1174.

109. Multicast Routing: Forsat. R., Haghighat, M., Mahdavi, M., "Harmony search based algorithms for bandwidth-delay-constrained least-cost multicast routing", Computer Communications, Elsevier

110. N.Dellart, J.Jeunet, A genetic algorithm to solve the general multi-level lot sizing problem with time varying costs. International journal of production Economics68 (2000)241-257.

111. N.P. Dellaert a, J. Jeunet b, Randomized multi-level lot-sizing heuristics for general product structures, European Journal of Operational Research 148 (2003) 211–228

112. N.P. Dellaert, J. Jeunet, Production, Manufacturing and Logistics Randomized multi-level lot-sizing heuristics for general product structures, European Journal of Operational Research 148 (2003) 211–228

113. Naderi, M. Mousakhani, M. Khalili, "Scheduling multi objective open shop scheduling using a hybrid immune algorithm", International Journal of Advanced Manufacturing Technology, pages 895-905.

114. Nhu Binh Ho, Joc Cing Tay, Edmund M.-K. Lai "An effective architecture for learning and evolving flexible job-shop schedules" European Journal of Operational Research, Volume 179, Issue 2, 1 June 2007, Pages 316-333.

115. Norman Sadeh, Katia Sycara, Yalin Xiong "Backtracking techniques for the job shop scheduling constraint satisfaction problem" Artificial Intelligence, Volume 76, Issues 1-2, July 1995, Pages 455-480.

116. Ong Y. S. and Lim M. H. and Zhu N. and Wong K. W. (2006). "Classification of Adaptive Memetic Algorithms: A Comparative Study". IEEE Transactions on Systems Man and Cybernetics -- Part B. 36 (1): 141.

117. Patrick R. Philipoom, Manoj K. Malhotra, Evaluation of scheduling rules with commensurate customer priorities in job shops Journal of Operations Management, Volume 13, Issue 3, October 1995, Pages 213-228.

118. Peng-Yeng Yin, Shiuh-Sheng Yu, Pei-Pei Wang, Yi-Te Wang, Task allocation for maximizing reliability of a distributed system using hybrid particle swarm optimization Journal of Systems and Software, Volume 80, Issue 5, May 2007, Pages 724-735.

119. Peter Brucker, Johann Hurink, Bernd Jurisch, Birgit Wöstmann "A branch & bound algorithm for the open-shop problem" Discrete Applied Mathematics, Volume 76, Issues 1-3, 13 June 1997, Pages 43-59.

120. Prithwish Chakraborty, Gaurab Ghosh Roy "On Population Variance and Explorative Power of Invasive Weed Optimization " World congress on nature and biologicaaly inspired computing, 2009

121. Qun Niu, Bin Jiao, Xingsheng Gu, Particle swarm optimization combined with genetic operators for job shop scheduling problem with fuzzy processing time Applied Mathematics and Computation, Volume 205, Issue 1, 1 November 2008, Pages 148-158.

122. R. M. Aiex, S. Binato, M. G. C. Resende "Parallel GRASP with path-relinking for job shop scheduling" Parallel Computing, Volume 29, Issue 4, April 2003, Pages 393-430.

123. R. Mallahzadeh,H. Oraizi and Z. Davoodi-Rad "Application of Invasive Weed Optimization technique for antenna configuration" Progress in Electro-magnetic Research, PIER 79, 2008Pages 137- 150

124. Rapeepan Pitakaso, Christian Almeder, A MAX-MIN Ant System for Unconstrained Multi-Level Lot-Sizing Problems, Industrial Engineering 51 (2005) 433–444

125. Regina Berretta, Luiz Fernando Rodriguez, A memetic algorithm for a multistage capacitated lot sizing problem, Int.j. Production Economics 87(2004)67-81.

126. Reza Tavakkoli-Moghaddam, Nima Safaei, Farrokh Sassani, A memetic algorithm for the flexible flow line scheduling problem with processor blocking Computers & Operations Research, Volume 36, Issue 2, February 2009, Pages 402-414.

127. Ruedee Masuchun, Wiboon Masuchun, Teerawat Thepmanee, Integrating m-Machine Scheduling into MRP, 2009 Fourth International Conference on Innovative Computing, Information and Control.

128. Runwei Cheng, Mitsuo Gen, Yasuhiro Tsujimura "A tutorial survey of job-shop scheduling problems using genetic algorithms, part II: hybrid genetic search strategies" Computers & Industrial Engineering, Volume 36, Issue 2, April 1999, Pages 343-364.

129. S. D. Müller, J. Marchetto, S. Airaghi, and P. Koumoutsakos, "Optimization based on bacterial chemotaxis," IEEE Transactions on Evolutionary Computation, vol. 6, no. 1, pp. 16–29, 2002.

130. S.Q. Liu, H.L. Ong, K.M. Ng "A fast tabu search algorithm for the group shop scheduling problem" Advances in Engineering Software, Volume 36, Issue 8, August 2005, Pages 533-539.

131. S.Q. Liu, H.L. Ong, K.M. Ng "Applying Tabu search to Job Shop Scheduling Problem" Annals of Operations research 41, 1993, Pages 231-252

132. S.Q. Liu, H.L. Ong, K.M. Ng "Metaheuristics for minimizing the makespan of the dynamic shop scheduling problem" Advances in Engineering Software, Volume 36, Issue 3, March 2005, Pages 199-205.

133. Sabuncuoglu, M. Bayiz, Job shop scheduling with beam search European Journal of Operational Research, Volume 118, Issue 2, 16 October 1999, Pages 390-412.

134. Satellite Heat Pipe Design: Geem, Z. W. and Hwangbo, H. "Application of Harmony Search to Multi-Objective Optimization for Satellite Heat Pipe Design", Proceedings of US-Korea Conference on Science, Technology, & Entrepreneurship (UKC 2006), CD-ROM, Teaneck, NJ, USA, Aug. 10-13, 2006.

135. Siddharth Pal, Anniruddha Basak and Swagatam Das "A Invasive Weed Optimization method based Multi-user detection for MC-CDMA Interference suppression over multiple-path fading channel " 978-422-6588-0/10/&25.00, IEEE,2010

136. Siddharth Pal, Anniruddha Basak and Swagatam Das "Circular Antenna Array synthesis with a differential Invasive Weed Optimization Algorithm"10th International Conference on Hybrid Intelligent systems ,2010

137. Silver, E. and Meal, H. (1973). A heuristic for selecting lot size requirements for the case of a deterministic time-varying demand rate and discrete opportunities for replenishment. Production and Inventory Management, 14(2):64–74.

138. Soil Stability Analysis: Cheng, Y. M., Li, L., Lansivaara, T., Chi, S. C. and Sun, Y. J. "An Improved Harmony Search Minimization Algorithm Using Different Slip Surface Generation Methods for Slope Stability Analysis", Engineering Optimization, 2008.

139. Staggemeier, A.T., et Clark, A.R.: A survey of lot-sizing and scheduling models, 23rd Annual Symposium of the Brazilian Operational Research Society, (2001) 603-617.

140. Structural Design: Lee, K. S. and Geem, Z. W. "A New Structural Optimization Method Based on the Harmony Search Algorithm", Computers & Structures, 2004.

141. Sylverin Kemmoé Tchomté, Michel Gourgand, Particle swarm optimization: A study of particle displacement for solving continuous and combinatorial optimization problems International Journal of Production Economics, Volume 121, Issue 1, September 2009, Pages 57-67.

142. T. S. Raghu, Chandrasekharan Rajendran, Combinatorial evaluation of six dispatching rules in a dynamic two-machine flow shop Omega, Volume 24, Issue 1, February 1996, Pages 73-81.

143. T. Stützle et H.H. Hoos, MAX MIN Ant System, Future Generation Computer Systems, volume 16, pages 889-914, 2000

144. Takeshi Yamada and Ryohei Nakano "Genetic Algorithm for Job-shop scheduling problem" Proceedings of Modern Hueristic for Decision support, pp.March 1997, Pages 67-81.

145. Takeshi Yamada and Ryohei Nakano" Job shop scheduling "Job Shop Scheduling, pp.IEEE Control engineering Services 55,pages 134-160

146. Tour Planning: Geem, Z. W., Tseng, C. -L., and Park, Y. "Harmony Search for Generalized Orienteering Problem: Best Touring in China", Lecture Notes in Computer Science, 2005.

147. Tsung-Lieh Lin, Shi-Jinn Horng, Tzong-Wann Kao, Yuan-Hsin Chen, Ray-Shine Run, Rong-Jian Chen, Jui-Lin Lai, I-Hong Kuo, An efficient job-shop scheduling algorithm based on particle swarm optimization Expert Systems with Applications, In Press, Corrected Proof, Available online 21 August 2009.

148. V. Granville, M. Krivanek, J.P. Rasson, "Simulated annealing: A proof of convergence". IEEE Transactions on Pattern Analysis and Machine Intelligence 16 (6): 652-656. June 1994.

149. Vehicle Routing: Geem, Z. W., Lee, K. S., and Park, Y. "Application of Harmony Search to Vehicle Routing", American Journal of Applied Sciences, 2005.

150. Visual Correspondence: J. Fourie, S. Mills and R. Green "Directed correspondence search: Finding feature correspondences in images using the Harmony Search algorithm", Image and Vision Computing New Zealand, 23-25 Nov. 2009. 24rd International Conference.

151. Visual Tracking: J. Fourie, S. Mills and R. Green ,"Visual tracking using the harmony search algorithm", Image and Vision Computing New Zealand, 2008. 23rd International Conference.

152. Voratas Kachitvichyanukul · Siriwan Sitthitham, "A two-stage genetic algorithm for multi-objective job shop scheduling problems, Journal of Intelligent Manufacturing, Springer, 2009.

153. W.W.Trigeiro, L.J.Thomas, J.O.McClain, Capacitated Lot Sizing with set up times, Management Science 35(1989) 353-366.

154. Wagner, H. and Whitin, T. (2004), Dynamic version of the economic lot size model. Management Science, 50(12):1770–1774.

155. Wang Shi-jin, Xi Li-feng, Zhou Bing-hai, Filtered-beam-search-based algorithm for dynamic rescheduling in FMS Robotics and Computer-Integrated Manufacturing, Volume 23, Issue 4, August 2007, Pages 457-468.

156. Xing-Quan Zuo, Yu-Shun Fan " Solving the Job Shop Scheduling Problem by an Immune Algorithm" Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, 18-21 August 2005.

157. Xueni Qiu · Henry Y. K. Lau, "An AIS-based hybrid algorithm for static job shop scheduling problem" Journal of Intelligent Manufacturing, 2019.

158. Yang S, Wang D (2001) ,"A new adaptive neural network and heuristics hybrid approach for job shop scheduling", Comput Oper Res 28:955–971.

159. Yi Han, Ikou Kaku, Jiafu Tang, Nico Dellaert, A Scatter Search Approach for Uncapacitated multilevel Lot-Sizing Problems, ICIC International c 2011 ISSN 1349-4198

160. Yi Hana, Jiafu Tanga, Iko Kakub, Lifeng Mua, Solving uncapacitated multilevel lot-sizing problems using a particle swarm optimization with flexible inertial weight, Computers and Mathematics with Applications 57 (2009) 1748_1755

161. Yuli Zhang ,Shiji Song ,Heming Zhang ,Cheng Wu , Wenjun Yin, A hybrid genetic algorithm for two-stage multi-item inventory system with stochastic demand, Neural Comput & Applic (2012) 21:1087–1098.

Biodata of author

Sudhir Kumar Mishra, Distinguished Scientist & Chief Controller Research & Development (BrahMos), DRDO, Ministry of Defence, Govt. of India & CEO & MD, BrahMos Aerospace.

Sudhir Kumar Mishra did his B.Tech from University of Jabalpur in Mechanical Engineering in the year 1982 and M.Tech. from IIT Madras in the year 1995 and currently pursuing PhD from NIT Warangal.

He is currently Working as Distinguished Scientist in DRDO and leading highly motivated R&D team to design, develop, test, produce and maintain, encompassing complete life cycle of world's fastest supersonic cruise missile BRAHMOS, for Indian Armed Forces. The BRAHMOS Missile is successfully developed and inducted into Indian Army, Navy and Air Force totaling to order value of Rs.27190 crores. BrahMos Aerospace has recorded annual turnover of Rs.2300 crores and have excellent profit margins. Since 01 Aug 2014 till date as Distinguished Scientist & Chief Controller Research & Development (BrahMos) & CEO&MD, BrahMos Aerospace and Chairman, BATL Thiruvananthapuram.

Leading highly motivated and technically competent team of more than 1500 Engineers, Scientists and Technicians located at New Delhi, Hyderabad, Nagpur and Thiruvananthapuram and have successfully managed smooth functioning of prestigious International Joint Venture Company BrahMos. His responsibilities include design, development, testing, production of BRAHMOS Supersonic Cruise Missile Systems.

****