

A Neural Network Guided Genetic Algorithm for Flexible Flow Shop Scheduling Problem with Sequence Dependent Setup Time

by

Syeda Manjia Tahsien

A Thesis

presented to

The University of Guelph

In partial fulfilment of requirements

for the degree of

Master of Applied Science

in

Engineering + AI

Guelph, Ontario, Canada

© Syeda Manjia Tahsien, December, 2020

ABSTRACT

A NEURAL NETWORK GUIDED GENETIC ALGORITHM FOR FLEXIBLE FLOW SHOP SCHEDULING PROBLEM WITH SEQUENCE DEPENDENT SETUP TIME

Syeda Manjia Tahsien
University of Guelph, 2020

Advisor:
Professor F.M. Defersha

This thesis presents a discriminating technique and clustering ordered permutation using Adaptive Resonance Theory (ART) and potential applications in the ART-guided Genetic Algorithm (GA). In this regard, we have introduced two novel techniques for converting ordered permutations to binary vectors to cluster them using ART. The proposed binary conversion methods are evaluated under varying parameters, and problem sizes with the performance analysis of ART-1 and Improved-ART-1. The numerical results indicate the superiority of one of the proposed binary conversion techniques over the other and Improved-ART-1 over ART-1. Finally, we develop Improved-ART-1 Neural Network guided GA to solve a flexible flow shop scheduling problem (FFSP) with sequence dependant setup time. Numerical examples show that ANN-guided GA outperforms the pure GA in solving several large size FFSP problems.

Keywords: *Adaptive Resonance Theory; ANN-Guided Genetic Algorithm; Binary Conversion Method; Flexible Flow Shop; Genetic Algorithm; Neural Network; Ordered Permutations.*

*At first, I would like to give my sincere and humble thanks to the Almighty, the
most gracious and the most merciful.*

*Dedicated to my parents, in-laws, my beloved husband, Raihan, and my sisters
who supported me from the very beginning to accomplish every success in this
journey.*

ACKNOWLEDGEMENTS

First and foremost, I would like to express my most sincere gratitude and appreciation to my wonderful and humble supervisor, Dr. Fantahun Defersha, whose support, guidance, and patience have been invaluable throughout my research and study time at the University of Guelph. I cannot thank him enough for the guidance, valuable time, and effort he spent teaching and supervising me during my research. Without his advice and encouragement, this MASc thesis would not have been at this stage and this shape and form. I would also like to extend my Thanks to my advisory committee member, Dr. S. Andrew Gadsden, for reviewing my thesis and guide me from time to time during my research and writing.

My most heartfelt thanks go to my beloved parents for their endless support, unwavering encouragement, and continuous pray. I am greatly indebted to them for the countless sacrifices they have made for me throughout my life. I would also like to express my special thank to my lovely husband, who is the most significant indirect contributor to this study period at the University of Guelph. He gave me tremendous support and encouragement during this period to complete my research. Besides, I want to express my gratitude to my in-laws for their support and motivation in this journey. Besides, I am deeply thankful to all the faculty members, staff, and labmates at the University of Guelph.

TABLE OF CONTENTS

ABSTRACT	ii
DEDICATION	iii
ACKNOWLEDGMENT	iv
LIST OF FIGURES	ix
LIST OF TABLES	xii
LIST OF SYMBOLS	xiv
LIST OF ACRONYMS	xvi
 1 Introduction	 1
1.1 Manufacturing Scheduling	4
1.1.1 Single Machine Scheduling Problem	5
1.1.2 Parallel Machine Scheduling Problem	6
1.1.3 Job Shop Scheduling Problem	6
1.1.4 Flow Shop Scheduling Problem	7
1.1.5 Open Shop Scheduling Problem	9
1.2 Scheduling Optimization Algorithms	9
1.2.1 Heuristic Algorithm	11
1.2.2 Metaheuristic Algorithm	12
1.2.3 Evolutionary Algorithm	13
1.3 Objectives	13
1.4 Contribution of the present research	14
1.5 Organization of this thesis	15
 2 Literature Review	 16
2.1 Introduction	16
2.2 GA and NN Approaches	17
2.3 GA and NN in Manufacturing Scheduling	20
2.3.1 NN in Manufacturing Scheduling	22

2.3.2	GA in Manufacturing Scheduling	25
2.3.3	Hybridization of GA	27
2.4	Research Motivation	32
3	Binary Conversion Methods and ART Neural Network	34
3.1	Introduction	34
3.2	Proposed Binary Conversion Methods	35
3.2.1	Method-1	35
3.2.2	Method-2	37
3.3	Adaptive Resonance Theory	38
3.3.1	Classification of ART	38
3.3.2	Advantages and limitation of ART	39
3.3.3	Appliations of ART	40
3.3.4	Architecture of ART	41
3.4	ART-1 and Improved ART-1 Learning Mechanism	42
3.4.1	ART-1 Learning Technique	43
3.4.1.1	ART-1 Learning Stages	46
3.4.1.2	ART-1 Learning Algorithm	49
3.4.2	Improved-ART-1 Learning Technique	50
3.5	Numerical Study and Performance Analysis	51
3.5.1	Experimental Data Generation	51
3.5.1.1	Misclassification	52
3.5.1.2	Homogeneity	53
3.5.1.3	Average Distance	53
3.5.2	Comparison between M1 and M2 with ART-1	53
3.5.3	Comparison between M1 and M2 with Improved-ART-1	56
3.5.4	Comparison between ART-1 and Improved-ART-1 with M1	59
3.5.5	Effect of Half-Width on Performance for Improved-ART-1 using M1	62

3.5.6	Effect of Degree of Perturbation on Improved ART-1 using M1	64
3.5.7	Effect of Vigilance Value on Improved-ART-1 with M1	67
3.6	Conclusion	70
4	ANN Guided Genetic Algorithm	71
4.1	Flexible Flow Shop Scheduling	72
4.2	Mathematical Formulation	73
4.2.1	Assumption and Problem Description	73
4.2.2	Notations	74
4.2.3	MILP Model	75
4.3	Pure Genetic Algorithm	78
4.3.1	Solution Representation	80
4.3.2	Fitness Evaluation	81
4.3.3	Genetic Operators	82
4.3.3.1	Selection Operator	83
4.3.3.2	Crossover Operator	84
4.3.3.3	Mutation Operator	85
4.4	Proposed ANN-guided GA	86
5	Numerical Investigation	89
5.1	Prototype Problem	90
5.2	Typical Solution	95
5.3	Solution of Prototype Problem (GA vs CPLEX)	98
5.4	GA and ANN-guided GA for Large Size Problems	101
6	Research Outline	110
6.1	Summary and Conclusion	110
6.2	Future Research and Recommendations	111

Bibliography**113**

LIST OF FIGURES

1.1	A basic structure of an ANN.	3
1.2	Schematic diagram of a parallel machine scheduling problem.	6
1.3	Schematic presentation of a simple job shop scheduling.	7
1.4	Schematic diagram of a simple flow shop scheduling problem.	8
1.5	A flow chart of different optimization algorithms.	10
1.6	Different algorithms used in optimization problem since 2013.	11
2.1	Statistical presentation in terms of publication number since 1985 till today on scheduling problems using GA, NN, and hybridization of GA with NN.	22
2.2	Number of publications over last three decades on job shop and flow shop scheduling problems that used GA and NN together.	23
2.3	Van diagram presenting a possible hybridization option for research. . .	28
3.1	Method-1 conversion process.	36
3.2	Method-2 conversion process.	37
3.3	Schematic presentation of basic architecture of an ART.	41
3.4	Basic architecture of ART-1 (Pandya and Macy, 1997).	43
3.5	Recognition layer of ART-1 (Pandya and Macy, 1997).	44
3.6	Comparison layer of ART-1 (Pandya and Macy, 1997).	45
3.7	Operation stage 1 of ART-1 where G1 is 1 (Pandya and Macy, 1997). .	47
3.8	Operation stage 2 of ART-1 where G1 is 0 (Pandya and Macy, 1997). .	47
3.9	Operation stage 3 of ART-1 (Pandya and Macy, 1997).	48
3.10	Operation stage 4 of ART-1 (Pandya and Macy, 1997).	49
3.11	Change in homogeneity along with cluster ID for M1 and M2 considering ART-1.	56
3.12	Average distance vs cluster ID for M1 and M2 considering ART-1. . . .	57

3.13	Change in homogeneity along with cluster ID for M-1 and M-2 of Improved-ART-1.	59
3.14	Change in average distance with cluster ID for M-1 and M-2 of Improved-ART-1.	60
3.15	Change in misclassification between two methods for ART1 and Improved-ART-1.	60
3.16	Comparison between ART-1 (M-1) and Improved-ART-1 (M-1).	61
3.17	Comparison between ART-1 and Improved-ART-1 with M1 at different number of seeds at vigilance value 0.5.	62
3.18	change in misclassification as a function of degree of perturbation for Improved-ART-1 using conversion methods M1.	66
3.19	Change Homogeneity and average distance with degree of perturbation for Improved-ART-1	67
3.20	Change in number of misclassification with vigilance parameter for Improved-ART-1 for 40 objects.	69
4.1	A basic flow chart of GA.	79
4.2	Schematic presentation of GA structure.	79
4.3	Solution representation.	81
4.4	Presentation of single point crossover-1.	85
4.5	An example of swap mutation.	85
4.6	An example of shift mutation.	86
4.7	Architecture of the Improved-ART-1 ANN-guided GA.	88
5.1	A typical solution representation for FFSP.	95
5.2	Gantt chart of machine scheduling for prototype problem solution.	99
5.3	Optimality graph for prototype problem using BC CPLEX solver.	100
5.4	Convergence graph for prototype problem using pure GA.	100
5.5	A typical convergence graphs (Blue - Worse Solution; Orange - Average; Gray - Good; Yellow - Best so far)	103

5.6	Comparison over multiple trails between GA and ANN-guided GA for problem 1.	104
5.7	Comparison over multiple trails between GA and ANN-guided GA for problem 2.	105
5.8	Comparison over multiple trails between GA and ANN-guided GA for problem 3.	105
5.9	Comparison over multiple trails between GA and ANN-guided GA for problem 4.	106
5.10	Comparison over multiple trails between GA and ANN-guided GA for problem 5.	106
5.11	Comparison over multiple trails between GA and ANN-guided GA for problem 6.	107
5.12	Formation of cluster by allocating the members based on the fitness value by GA (a), (c), (e) and ANN-guided GA (b), (d), (f) for problems 1, 3, and 5 respectively mentioned in Table 5.8.	108
5.13	Allocation of population with and without intervention of ANN. (a), (c), (e) are the results of pure GA and (b), (d), (f) are for ANN-guided GA considering problems 4, 5, and 6 respectively mentioned in Table 5.8; the legends represent the cluster id.	109

LIST OF TABLES

1.1	Variants of shop scheduling in manufacturing industry (Azadeh <i>et al.</i> , 2019; Bhatt and Chauhan, 2016; Rahmani Hosseinabadi <i>et al.</i> , 2019).	5
2.1	List of NN approaches used for different applications (Mehta, 2019; Team, 2020; Dagli and Sittisathanchai, 1995; Grossberg, 2013).	17
2.2	Advantages and disadvantages of GA and NN approach (Ansari and Bakar, 2015; Agarwal <i>et al.</i> , 2010)	21
3.1	Performance Analysis for ART-1 considering M1.	54
3.2	Performance Analysis for ART-1 considering M2.	55
3.3	Performance Analysis for Improved-ART-1 considering M1	57
3.4	Performance Analysis for Improved-ART-1 considering M2	58
3.5	Comparison between ART-1 and Improved-ART-1 considering different problem sizes. here, C= no. of cluster, M = misclassification, H= homogeneity, Mi= minimum, Mx= maximum, A= avergae, D= distance	63
3.6	Performance analysis for Improved-ART-1with binary conversion M1 considering different problem sizes at various half widths.	65
3.7	Performance analysis for Improved-ART-1 (M1) considering 40 objects for different vigilance value.	68
3.8	Performance analysis for ART-1 (M1) considering 40 objects for different vigilance value.	68
5.1	A small example of case study consists of two machines and four stages.	90
5.2	The case study includes eight jobs with different batch sizes and considered processing time for each eligible machine.	92
5.3	Setup time for each eligible machine for each job (Continued on next page).	93
5.4	continued from previous page Table 5.3.	94

5.5	Partial illustration of GA decoding steps for the solution representation in Fig. 5.1.	96
5.6	Continuation of Table 5.5 for GA decoding steps of job 8 ($n=8$).	97
5.7	Completion time of each job that is processed by each machine at four stages.	98
5.8	Feautres of considered large size problems.	101

LIST OF SYMBOLS

A	Average
$A_{n,i}$	A binary data equal to 1 if setup of job n in stage i is attached (non-anticipatory), or 0 if this setup is detached setup (anticipatory)
\mathbf{B}_j	Weight vector
$B_{n,i}$	A binary data equal to 1 if job n needs processing in stage i , otherwise 0
b_{ij}	Bottom up weights
\mathbf{C}	Output component vector
C_i	Cluster id
c_i	Output of i^{th} neuron in the comparison layer
c_{max}	Makespan of the schedule
$c_{n,i}$	Completion time of the job n from stage i
$\hat{c}_{r,m,i}$	Completion time of the r^{th} run of machine m in stage i
D	Distance
D_{p,C_i}	Permutations p from the cluster centroid C_i
$D_{p,C_{i'}}$	Permutation p to any other cluster
$D_{n,m,i}$	A binary data equal to 1 if job n can be processed on machine m in stage i , otherwise 0
F_1	Input/computational unit
F_2	Clustering/recognition unit
f	Step function
$F_{m,i}$	The release date of machine m in stage i
G	Gain control unit

$G2$	Gain-2
$G1$	Gain-1
H	Homogeneity
I	Total number of stages
j	Recognition layer neuron
k	Processing time
M	Misclassification
Mi	Minimum
Mx	Maximum
M_i	Number of machines
n	Number of jobs
N	Total number of jobs
o	Operation
$O_{n,p}$	Location of object- n in permutation p
P_i	Input from recognition layer
P_n	A set of pairs of stages
Q_n	Batch size of job n
\mathbf{R}	Feedback vector
$R_{m,i}$	Number of maximum production runs of machine m in stage i
r_j	Binary value
s	Location of the job
\mathbf{S}	Similarity ratio
$S_{m,i,n,p}$	Setup time on machine m in stage i for processing job n following the processing of job p on this machine
$S_{m,i,n,0}$	Setup time on machine m in stage i for processing job n and if the operation is the first to be processed

T_j	Wweight vector in the comparison layer
t_{ij}	Top down weights
$T_{n,m,i}$	Total processing time for one unit of job n on machine m in stage i
X	Number of seeds
$x_{r,m,i,n}$	Binary variable which takes the value 1 if the r^{th} run on machine m in stage i is for the job n , 0 otherwise,
Z	Defines the objectives
$z_{r,m,i}$	A binary variable which equal to 1 if the r^{th} potential run of machine m in stage i has been assigned a job to process, 0 otherwise
Ω	Large positive number
ψ_i	Net output of the recognition layer
Π_p	Cartesian coordinate of permutation p
X	Input pattern
ρ	Vigilance threshold

LIST OF ACRONYMS

AI	Artificial Intelligence
Aug-NN	Augmented NN
ANN	Artificial Neural Network
ART	Adaptive Resonance Theory
BC	Brand-and-Cut
BPNN	Back-Propagation Neural Network
CPS	Cyber Physics System
DE	Differential Evolution
EDA	Estimation of Distribution Algorithm
ES	Evolutionary Strategy
FFSP	Flexible flow shop scheduling problem
FSP	Flow shop scheduling problem
GA	Genetic Algorithm
GP	Genetic Programming
GWO	Grey Wolf Optimization
HW	Half Width
HFS	Hybrid Flow Shop Scheduling
IOAM	Intelligent Operations Assignment Mutation
IoT	Internet of Things
JSP/JSSP	Job Shop Scheduling Problem
LBRR	Lower Bound-Base Bias Roulette
M1	Method-1
M2	Method-2

ML	Machine Learning
MILP	Mixed-Integer Linear Programming
MOEA	Multiobjective Evolutionary Algorithms
MOEDA	Multi-Objective Estimation Distribution Algorithm
NE	Not Eligible
NEH	Nawaz-Enscore-Ham
NLP	Natural Language Processing
NN	Neural Network
NR	Not Required
NSGA-II	Non-Dominated Sorting Genetic Algorithm
OSSP	Open Shop Scheduling Problem
OSSM	Operations Sequence Shift Mutation
PMSP	Parallel Machine Scheduling Problem
PSO	Particle Swarm Optimization
ROAM	Random Operation Assignment Mutation
SA	Simulated Annealing
SFLA	Shuffled Frog-Leaping Algorithm
SMSP	Single machine scheduling problem
SPC	Single-Point Crossover
SPT	Shortest Processing Time Heuristic
SWCM	Solution Warehouse/Cluster Management
VS	Virtual Savant

Chapter 1

Introduction

Over the years, various sophisticated tools have been developed in order to help the human race. Digital computers and smart machines are the tools among those that have been dedicated to making human life easier and more comfortable. Now, we are more interested in making computers which can think and act like us, which indicates computers will be given intelligence that will artificially mimic our thoughts in its acts. John McCarthy has introduced the term “Artificial Intelligence (AI)” in 1956, and by using these words, he wanted to indicate a smart tool which is a combination of science and engineering and will be used for the betterment of the people. Since then, the research in AI has been flourishing at a rocket speed in every sector of our daily life. In simple words, AI is the branch of that science which mimics human intelligence in a machine. There are a couple of reasons for which AI has been adopted in almost every field by us, and these include:

- Provide better results.
- Create new opportunities for existing products.
- Optimize the system’s operation.

- Creates competition among developers, for which business becomes more dynamic.
- Reduce operational conflicts and speed up the operational process.

AI can be classified into two types. Type 1 includes narrow/weak AI and strong AI, whereas type 2 consists of limited memory, reactive machines, the theory of mind, and self-awareness. However, the word AI does not refer to any specific methodology or technique. AI can be compared as a big umbrella which covers machine learning (ML) (e.g., deep learning, supervised and unsupervised learning, predictive analytics, etc.), natural language processing (NLP) (e.g., translation, classification, clustering, information extraction, etc.), expert systems, vision, robotics. Every category is used in different applications using various algorithms and intelligent systems such as: the neural network (NN), Evolutionary algorithms, simulated annealing, computational intelligence, fuzzy logic, etc. In this thesis, we have studied one of the kinds of NN.

The NN concept comes from our brain where billions of neurons are interconnected and can handle difficult and complex tasks such as movement and control of different bodies, vision and face recognition, planning for body motion, etc. This human brain's neuron network has inspired the researchers to develop an artificial neural network (ANN) ([Gruau *et al.*, 1996](#); [de Garis, 1994](#)). ANN represents the models with nonlinear statistical data that mimic the role of the human brain's neurons ([Jain *et al.*, 1996](#)). McCulloch and Pitts initially investigated ANN in the 1940s ([McCulloch and Pitts, 1943](#)). The basic ANN model includes three layers, i.e., the input layer, hidden layer, and output layer. Like the human brain, neurons in ANN take the input and passes the information to another neuron. These neurons are connected by links, and each link has a weight to control the signal between the nodes/neurons. Moreover, these neurons are

capable of learning where they can change their weights. Figure 1.1 illustrates a basic structure of an ANN (Tahsien *et al.*, 2020).

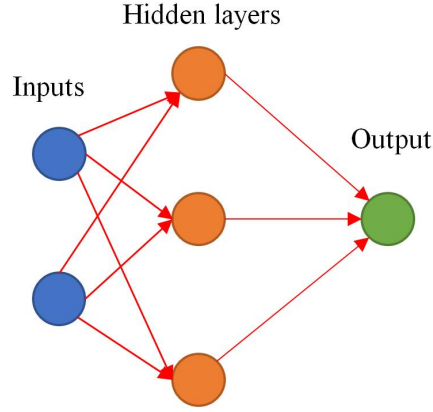


Figure 1.1: A basic structure of an ANN.

Over the last three decades, ANN has been adopted in engineering, automotive, medicine, economics, aerospace, energy sector, and many other fields (Yang *et al.*, 2008; Kumar *et al.*, 2013; Alippi *et al.*, 2003). Additionally, ANN has also attracted the manufacturing engineers' attention for resolving the scheduling problems (Hopfield and Tank, 1985; Agarwal *et al.*, 2006; Kasap and Agarwal, 2012; Huang and Gao, 2020). Research on scheduling is one of the oldest disciplines in the manufacturing industry, started in the 1950s (Yu and Liang, 2001). Since then, mathematical modeling, dynamic programming, branch, and bound method have been studied for manufacturing scheduling problems to optimize the outcome. On the other hand, this manufacturing scheduling becomes a young discipline because of new production and management technologies in this twenty-first century, especially in the Industry 4.0 (Zhang *et al.*, 2019). Traditional scheduling problems are considered centralized or some semi-distributed scheduling. In contrast, Industry 4.0 deals with advanced-smart distribution and optimization manufacturing technologies such as Digital Twin, Big Data, Internet of Things (IoT), Mass Customization, Cyber Physics System (CPS), Artificial

Intelligence (AI), Cloud Computing, Deep Learning, etc. Considering the revolution of Industry 4.0, the Genetic Algorithm (GA) and Neural Network (NN) have gained much attention in manufacturing scheduling problems. Therefore, this thesis's primary research has been focused on GA, NN, and hybridization of these two approaches in a manufacturing scheduling problem.

1.1. Manufacturing Scheduling

Scheduling generally refers to determining the completion time, date, and way of completing the task where all operations have been finished within the shortest possible time. The need for scheduling becomes essential in diverse fields such as manufacturing industries, supply chain processes, logistics, management, etc. Machines and operations are the two main resources of the manufacturing environment. It is necessary to link these two resources to get the optimum results from the production. Therefore, scheduling is one of the most critical factors in managing and planning the manufacturing process to determine a process that optimizes the production outcomes. Machine scheduling problem typically consists of operation sequencing and allocation problem, and this machine scheduling problem can be classified as (i) machine environment (shop scheduling); (ii) job characteristics; (iii) performance measures. The following subsections present a detailed description of the machine environment/shop scheduling types where our thesis work mainly evolves the flexible flow shop scheduling problem. Table 1.1 shows different types of shops that are considered in manufacturing scheduling.

Table 1.1: Variants of shop scheduling in manufacturing industry ([Azadeh *et al.*, 2019](#); [Bhatt and Chauhan, 2016](#); [Rahmani Hosseinabadi *et al.*, 2019](#)).

Shop types	Features
Single machine	One at a time
Parallel machine	Identical (Total identical and partial)
	Uniform
	Unrelated
Job shop	Multi-direction
Flow shop	Uni-direction
Open shop	No direction

The scheduling problems in manufacturing are classified into four categories, which are commonly used in theoretical computer science. “P” presents easy problems, “NP” represents medium, “NP-complete” presents hard, and “NP-hard” defines the hardest and complex problems.

1.1.1. Single Machine Scheduling Problem

Single machine is one of the most commonly researched problems in the classical scheduling problem due to its extensive range of applications in the real world ([Lu *et al.*, 2014](#)). Single machine scheduling problem (SMSP) refers to the process where a group of assigned tasks is performed in a single machine. The initial work of SMSP was accomplished by Jackson in the 1950’s ([Jackson, 1955](#); [Smith, 1956](#)). A detailed survey has been conducted on the theory and various applications of single machine scheduling problem by [Pinedo \(2012\)](#), [Abdul-Razaq *et al.* \(1990\)](#), and [Yen and Wan \(2003\)](#). When the assigned work becomes complicated, the entire single machine job is further subcategorized into several single machine jobs.

1.1.2. Parallel Machine Scheduling Problem

The parallel machine scheduling problem (PMSP) is defined as assigning several tasks/jobs to a number of parallel machines to get the optimum outcome. Theoretically, parallel machine is a complex form of single machine problem and a special scheduling problem of flexible flow shop. The application area of PMSP has a wide range of application areas, e.g., semiconductor area (Chien and Chen, 2007) and electronic manufacturing industries (Kaczmarczyk, 2011). PMSP can be further divided into identical, uniform, and unrelated parallel machines. Figure 1.2 illustrates a simple schematic diagram of PSMP.

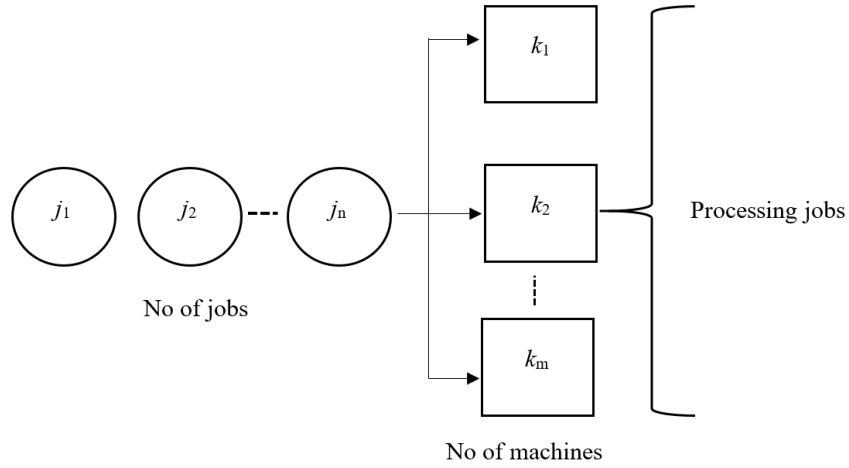


Figure 1.2: Schematic diagram of a parallel machine scheduling problem.

1.1.3. Job Shop Scheduling Problem

Job shop scheduling problem (JSP/JSSP) is one of the kinds of NP-hard optimization problems where a set of various jobs consisting of different operations are processed in machines at a particular time to minimize the makespan (overall completion time is known as makespan). In job shop scheduling, the machines' sequence for processing each job is different where there are no specific first and

final operations for each job. Figure 1.3 shows a general workflow of a job shop scheduling problem.

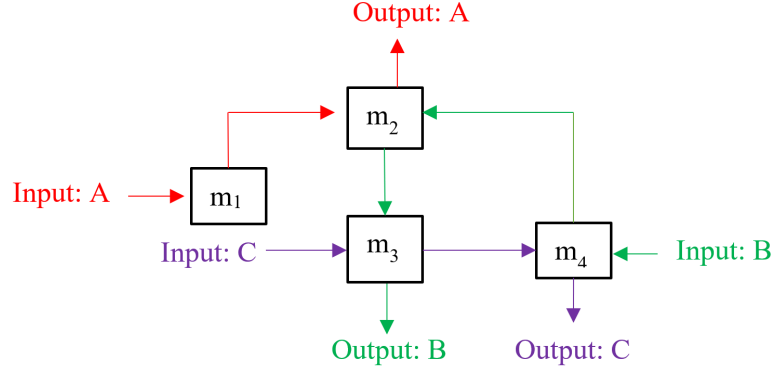


Figure 1.3: Schematic presentation of a simple job shop scheduling.

The commonly used JSP type is known as flexible JSP (FJSP), which is the complex and more advanced form of JSP. Recently, most of the optimization scheduling algorithms target JSP and FJSP, though their structure is different from the real-world industrial system. It is found from the literature that [Baker and Trietsch \(2009\)](#) introduced the FJSP for the optimization of the manufacturing scheduling problem.

1.1.4. Flow Shop Scheduling Problem

Flow shop is a unidirectional model where each given job is processed in a set of machines in identical order. In flow shop scheduling, each job can be processed at most by a machine, and each machine can handle only one job at a given time. Flow shop has been commonly used in mechanical manufacturing and industrial productions where jobs need to be processed incessantly in machines at a time in series ([Grabowski and Pempera, 2007](#)). Flow shop is a typical NP-hard type problem where it is difficult to find the global optima at polynomial time. Figure 1.4 illustrates a simple flow shop scheduling diagram.

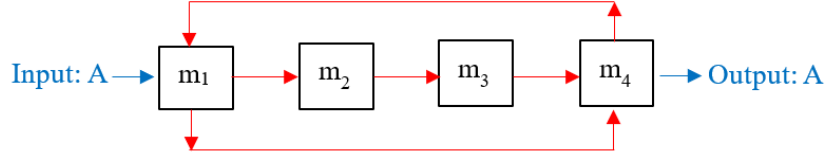


Figure 1.4: Schematic diagram of a simple flow shop scheduling problem.

Since 1954, researchers have been investigating flow shop scheduling problems (FSP) in real-world systems ([Johnson, 1954](#)). Though the mechanism of flow shop scheduling seems like an assembly line; however, there is a difference between them. Firstly, the assembly line only deals with a standard product, whereas FSP can process multiple jobs in multiple machines at a time. Secondly, it is necessary for a product to go through all the machines in the assembly line, whereas a job does not have to visit all the machines in FSP. Lastly, a machine can handle a job independently in FSP without depending on the previous stage, whereas one machine depends on the previous machine in assembly line scheduling.

The classical flow shop is known as a flexible flow shop (FFS/HFS). FFS deals with multiple workstations at one stage to minimize the makespan. FFS is widely considered in the electronics industry, chemical industry, automobiles sectors, etc. The number of available workstations at each stage is the same in both flow shop and flexible flow shop, where this limitation can be avoided using hybrid flow shop scheduling (HFS) ([Agarwal *et al.*, 2011](#)). In HFS, there are series of operation stages where jobs can be handled parallelly by multiple machines at a time.

1.1.5. Open Shop Scheduling Problem

Open shop scheduling problem (OSSP) is a non-directional NP-hard problem where each given set of tasks must be processed at a given period by a specific machine at a time. However, there is no operation sequence for the machines that are determined arbitrarily, i.e., each machine can handle one job at a time, and each job cannot be processed by more than one machine at a time. The aim of using such technique is to determine the schedule of the machine where it will be found the starting time of each job, which will reduce the overall job completion time. Gonzalez and Sahni ([Gonzalez and Sahni, 1976](#)) introduced this OSSP in industrial applications. For example, consider there is n number of jobs which will be processed by m number of machines. If all the n number of jobs are handled in a fixed route by m number of machines, it will be a flow shop scheduling problem. Moreover, if each job is handled by a specific route by m number of machines, it will be a job shop type problem. Now, if these job processing routes are not deterministic, then this is called OSSP.

1.2. Scheduling Optimization Algorithms

Optimization can be defined in various ways in different application fields. However, optimization is generally defined as a technique to use the system resources in order to identify the best outcomes in an efficient and effective way. Literature says that optimization techniques utilize different algorithms using accurate, effective, and real time information from surroundings to speed up the process of decision making and to ensure the best outcome ([Johnson *et al.*, 1993](#); [Coello, 1999](#)). Optimization includes two most important components i.e., analysis and modeling of a given problem. Since last century, researchers have been focusing on modeling for optimization techniques in real world. However, over the last three decades, researchers are also considering different algorithms and analysis

methods in this regard (Fu, 1994). Algorithms that are used in optimization techniques are known as optimization algorithms. Figure 1.5 illustrates a general classification of optimization algorithms.

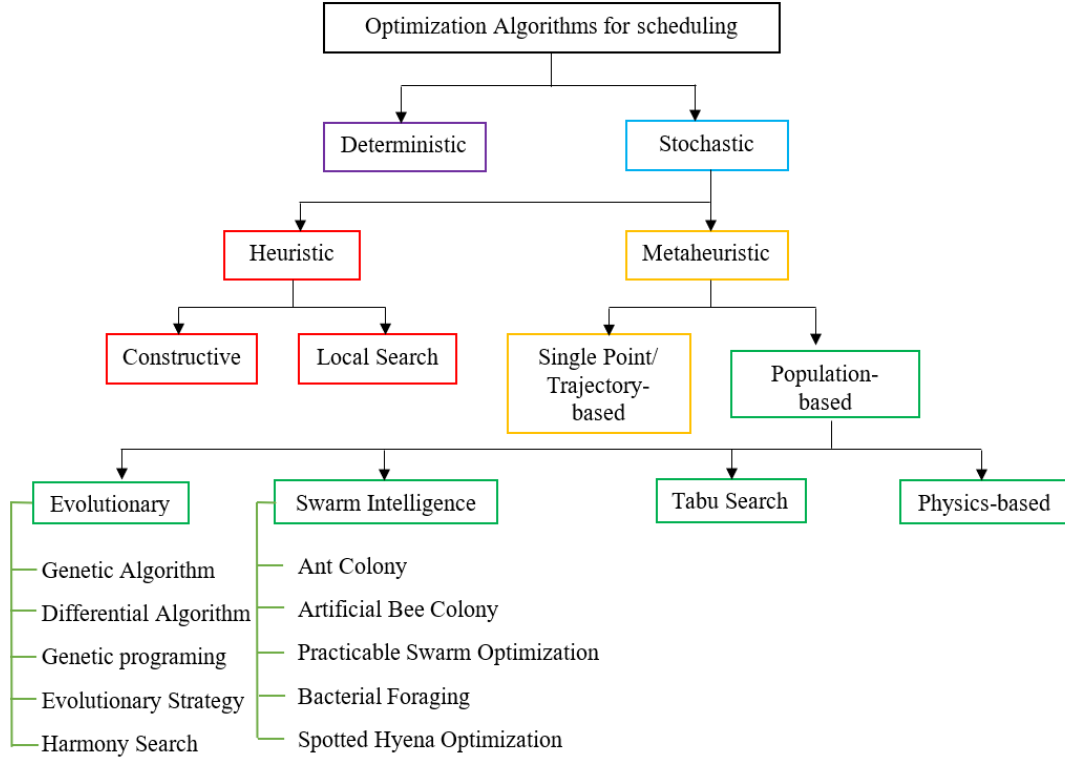


Figure 1.5: A flow chart of different optimization algorithms.

Optimization algorithms are typically classified into deterministic and stochastic algorithms. Deterministic algorithms deal with the global solution using a finite number of steps to optimize the outcomes of a problem, whereas the stochastic method is used for global optimization problems with bound-constrained and unconstrained. Stochastic methods are incorporated in order to find only one global solution, and it avoids repetitive the same outcome from global solutions. When clustering is required for any problem set, the stochastic methods become the best fit for the optimization technique over the deterministic methods. The stochastic algorithm is further divided into two main categories, such as heuristic and metaheuristic. The following subsection will describe these two stochastic

types of algorithms and their classification. Besides, Figure 1.6 shows the trend of using different algorithms as optimization techniques since 2013. In this figure, GA represents genetic algorithm; NN is the neural network; SFLA is shuffled frog-leaping algorithm; GWO stands for grey wolf optimization; PWSW presents particle swarm optimization; SA is simulated annealing; MOEA is multiobjective evolutionary algorithms; NSGA-II stands for non-dominated sorting genetic algorithm (data taken from different sources, e.g., Engineering village, IEEE Xplore, Springer, and others).

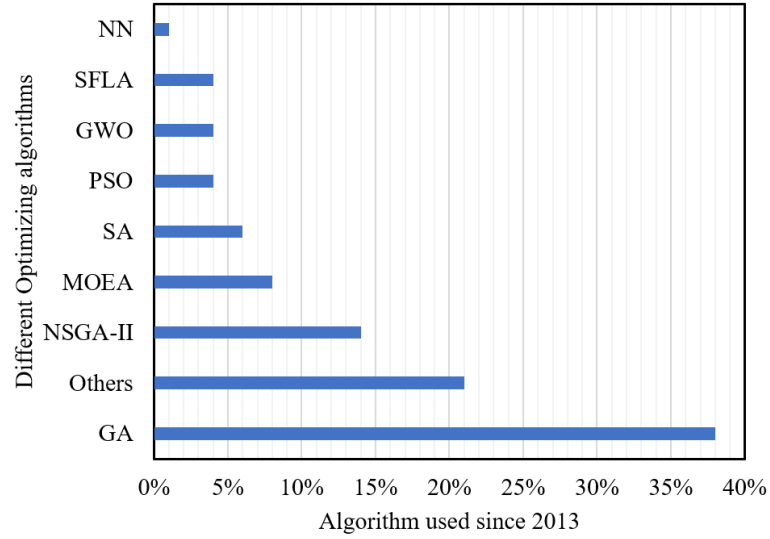


Figure 1.6: Different algorithms used in optimization problem since 2013.

1.2.1. Heuristic Algorithm

The heuristic is one of the kinds of stochastic algorithms widely used to find short term solutions for scheduling problems. The main advantage of adopting a heuristic algorithm in scheduling problems is that it can give a fast and inexpensive solution to the issue (Taskin, 2019; Frankovič and Budinská, 2000). Moreover, it is easy to understand and implement in the process. However, sometimes this quick response is not practical to use in the real field (Frankovič and Budinská,

2000).

1.2.2. Metaheuristic Algorithm

Over the last three decades, various optimization algorithms have been introduced considering the basic guidelines of heuristic in order to find a fast and feasible solution for scheduling problems, which are commonly known as metaheuristic algorithms. It is the up-gradation of heuristic algorithms, which is a repetitive method to find the optimal solution, which is also called the modern heuristic method. The common features of the metaheuristic algorithms are:

- Metaheuristic algorithms are considered to find the near-optimal solution of any problem.
- Metaheuristic algorithms are not for any specific problem.
- Metaheuristic algorithms are the stochastic process.
- Metaheuristic algorithms are constituted in order to use in local search as well all in any complex learning systems.
- Metaheuristics can direct the search algorithms into any favorable regions.
- The results of metaheuristics can be easily matched with the simulated results, making it more feasible for any scheduling problem to validate.

Metaheuristic algorithms can be classified into single-point/trajectory-based and population-based algorithms (see Fig.1.5). Single point/trajectory-based metaheuristic algorithm falls into Hill Climbing, Simulated Annealing, Tabu Search algorithms. Population-based metaheuristic algorithms are further divided into Evolutionary algorithms, swarm algorithms, Neural networks, Fuzzy programming, etc. In this thesis, the research is focused on the Genetic algorithm (GA), which is a kind of Evolutionary metaheuristic algorithm.

1.2.3. Evolutionary Algorithm

Evolutionary algorithms are stochastic meta-heuristic algorithm which is encouraged by the biological evolution. The main idea behind these algorithms is to utilize the randomness with mating characteristics and transmutation to evolve our required optimal solution. When there is a bundle of solutions and a need to find a near-optimal solution, then evolutionary algorithms come first in the race to be considered. Evolutionary algorithms are widely used in scheduling problems, image processing, knapsack problems, routing, etc. Evolutionary algorithms include Genetic algorithm (GA), Differential Evolution (DE), Evolutionary Strategy (ES), Genetic Programming (GP), Harmony search. Among different evolutionary algorithms, GA has attracted the researchers most, which is also noted from Fig.1.5. Therefore, it motivates us to do a literature survey and research on GA techniques in the proposed hybrid approach for manufacturing scheduling problems.

1.3. Objectives

Manufacturing scheduling problems are very complex and diverse in nature, and there is no one method that could solve them efficiently. Machine learning (ML) based genetic algorithm can be a practical approach to handle the complexity of scheduling problems by utilizing the learning ability of ML and the global search capability of genetic algorithm. A genetic algorithm (GA) is an optimization technique that works based on the principles of evolution and natural selection. It has been applied to solve a wide range of problems, both having continuous and/or discrete decision variables. The selection process in a GA is accomplished by assigning a selection probability to each individual solution proportional to the measure of goodness of the solution (fitness function). However, in discrete

optimization, the goodness of a solution can be significantly altered and deteriorate by a slight perturbation of the variables. In that case, a solution that can be easily improved to provide a good solution can be discarded by the GA. Hence, discriminating solutions merely based on fitness function may result in a poor convergence of the GA as many solutions that can be easily altered to good solutions can potentially be rejected by the search process. In this scenario, characterizing solutions based on genotype information, in addition to the phenotype, can be a promising approach. The overall research focus is to integrate the ANN, especially adaptive resonance theory (ART), with GA in solving the selected scheduling problem with the objective of improving the convergence behavior of the genetic algorithm and the quality of the final solution. The main objective of using ART as NN in the current work are: (i) ART does not forget the previous result, i.e., ART can adopt and learn well at any stage of operation without forgetting old result, which is very useful for any applications; (ii) it is always open for new learning; (iii) Other NNs need time-consuming training based on advanced knowledge, and it is tough to introduce new data once its training is completed which can be overcome by considering ART.

1.4. Contribution of the present research

This research work introduces Improved ART-1 neural network-guided GA considering proposed binary conversion methods in a flexible flow shop scheduling problem and reduces the makespan. ART neural network has been used in many applications due to its fast-adaptable learning process and stable operations. In this work, we have presented a discriminating technique and clustering ordered permutation using ART-1 and Improved-ART-1. In the process, we developed novel techniques for converting ordered permutations to binary vectors to cluster them using ART. The performances of ART-1 and Improved-ART-1 have been

investigated, and the proposed binary conversion methods were evaluated under varying parameters and problem sizes. Three performance indicators, i.e., misclassification, cluster homogeneity, and average distance, are considered in the analysis. The numerical results indicate the superiority of one of the proposed binary conversion techniques over the other and Improved-ART-1 over ART-1. Moreover, potential applications of the proposed technique in developing ANN guided metaheuristics to solve problems whose solutions are ordered permutations are discussed. Different small and large size problems in solving flexible flow shop scheduling using ANN-guided GA is also studied and analyzed to compare with pure GA.

1.5. Organization of this thesis

The thesis is organized into six chapters. The rest of the thesis proceeds as follows. Chapter 2 presents the literature survey on metaheuristic algorithms mainly focused on GA, NN, and GA's hybridization. In Chapter 3, adaptive resonance theory (ART), an artificial neural network, has been discussed and introduced two binary conversion methods for ordered permutations used for clustering in ART. Chapter 4 discusses the detailed mechanism of GA that is going to be used in the case study and combining the architecture of GA with ART. In Chapter 5, numerical examples of ART neural network-guided GA are provided, and a comparative study has been illustrated. At last, the conclusion and future work have been presented in Chapter 6.

Chapter 2

Literature Review

2.1. Introduction

Over the last two decades, noticeable efforts have been accomplished to investigate the research possibility of using Genetic algorithms (GA) and other machine learning techniques in manufacturing scheduling problems. Furthermore, the hybridization of GA with another learning method has been proved as a potential alternative to traditional optimizations techniques, especially for complex scheduling problems. Therefore, in this chapter, a comprehensive literature review has been presented on manufacturing scheduling, considering GA as a metaheuristic algorithm and Neural Network (NN) as an artificial intelligence learning technique. Moreover, the literature survey has been more focused on the hybridization of GA with the NN approach for various manufacturing scheduling problems.

This chapter is organized as follows: an overview of GA and NN techniques presents in section 2.2; a comprehensive literature survey on GA, NN and combined approach of GA and NN in different manufacturing scheduling problems present in section 2.3; at last, section 2.4 represents the research motivation of

the thesis.

2.2. GA and NN Approaches

Genetic algorithm (GA) is one of the powerful search-based optimization tools which mimics the principle of genetics inspired by the natural selection process to find the optimal solution. GA is a branch of evolutionary computational algorithms. Jhon Holland is the father of GA, who introduced this algorithm in the 1960s based on the evolutionary theory ([Sadeghi *et al.*, 2014](#)). A neural network (NN) is generally referred to as an artificial neural network (ANN) inspired by biological nerve systems. NN consists of several neurons which are interconnected like human brains, weights, and propagation function ([Awodele and Jegede, 2009](#)). A list of various NN approaches with their architecture and application areas is presented in Table 2.1, and advantages and disadvantages are summarized in Table 2.2.

Table 2.1: List of NN approaches used for different applications ([Mehta, 2019](#); [Team, 2020](#); [Dagli and Sittisathanchai, 1995](#); [Grossberg, 2013](#)).

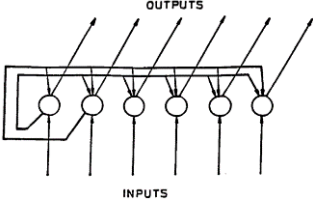
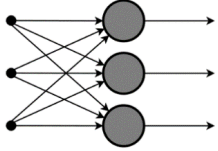
Types of NN	Schematic Diagram	Remarks
Hopfield		Image recognition and restoration
Feedforward		Simplest type of NN
Continued on next page		

Table 2.1 – continued from previous page

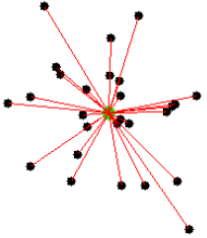
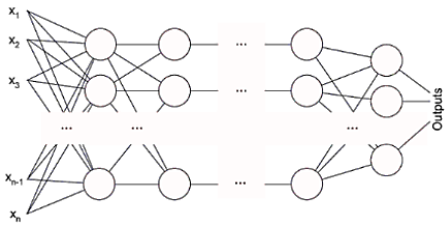
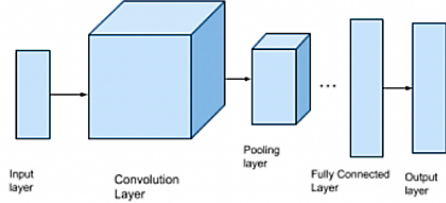
Types of NN	Schematic Diagram	Remarks
		Used in computer vision, face recognition, speech recognition
Radial Basis Function		Widely used in power restoration system
Multilayer Perception		More than three layers Used in machine translation and speech recognition technologies, complex classification
Convolutional		Shows good results in paraphrase and parsing detection
Continued on next page		

Table 2.1 – continued from previous page

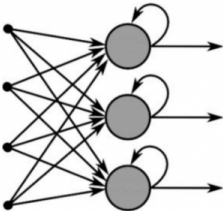
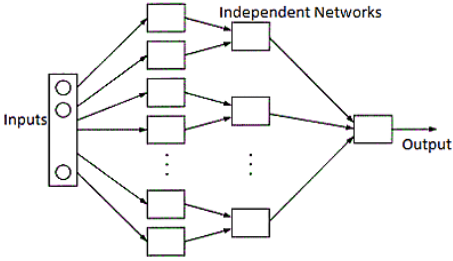
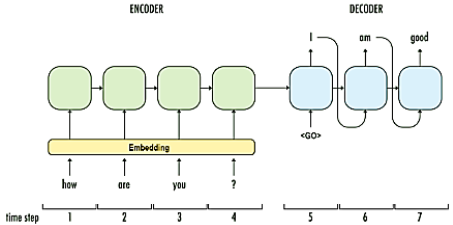
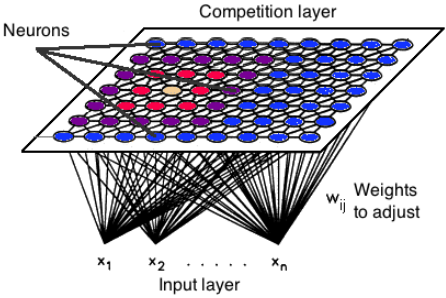
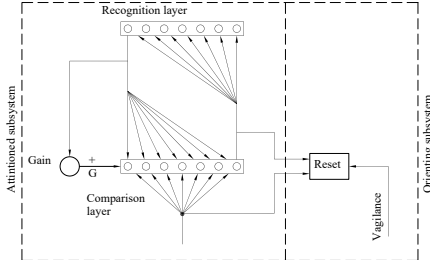
Types of NN	Schematic Diagram	Remarks
		Used in image processing, speech recognition, computer vision, machine translation
Recurrent		Used in image analysis, text to speech processing, translation, sentiment analysis
Modular		Used in stock market prediction system, character recognition
Sequence to Sequence		Used in chatbot and machine translation
Continued on next page		

Table 2.1 – continued from previous page

Types of NN	Schematic Diagram	Remarks
Kohonen Self Organizing		Used mainly in pattern recognition in the data
Adaptive Resonance Theory (ART)		Used in face recognition, signature verification, mobile robot control, remote sensing, airplane design, autonomous adaptive robot control, target recognition, medical diagnosis, face recognition, land cover classification, fitness evaluation etc

2.3. GA and NN in Manufacturing Scheduling

This section deals with the literature survey on GA, NN, and GA's hybridization with NN. In this context, it is found that the number of publications on GA in scheduling problems is increasing rapidly, whereas the incremental rate of NN and hybridized methods are not that significant (see Fig. 2.1). However, the statistics

Table 2.2: Advantages and disadvantages of GA and NN approach ([Ansari and Bakar, 2015](#); [Agarwal *et al.*, 2010](#))

Approach	Advantages	Disadvantages
GA	Can solve problems with multiple solutions (NP hard), specially for global search	Time consuming method and needs a lot of iterations
	Easy to understand and easy to apply to any complex problem	Cannot store the previous results
	Can handle wider class simultaneously due to its robust structure	Sometimes difficult to find the global optimum solution
	Capable of finding optimum solution using chromosome encoding.	Comparatively less effective method for local search
NN	Good at local search	Comparatively less effective at global search
	Depends on heuristics	Slow but needs comparatively less iterations than GA
	By nature, its operation is parallel which makes it more effective	Needs proper training for compiling the problem and find best results
	Can work efficiently even if one neuron fails in the middle of the iteration	Once it is trained, it cannot be trained during the process
	Has adaptive learning potentiality and generalize the problem	

indicate that the research interest in considering a hybrid approach raises over the last decade, which can be a great opportunity for doing research more. Fig. 2.2 shows the number of published articles on the hybridization of GA and NN in job shop and flow shop scheduling problems where job shop scheduling problem is the leading research area.

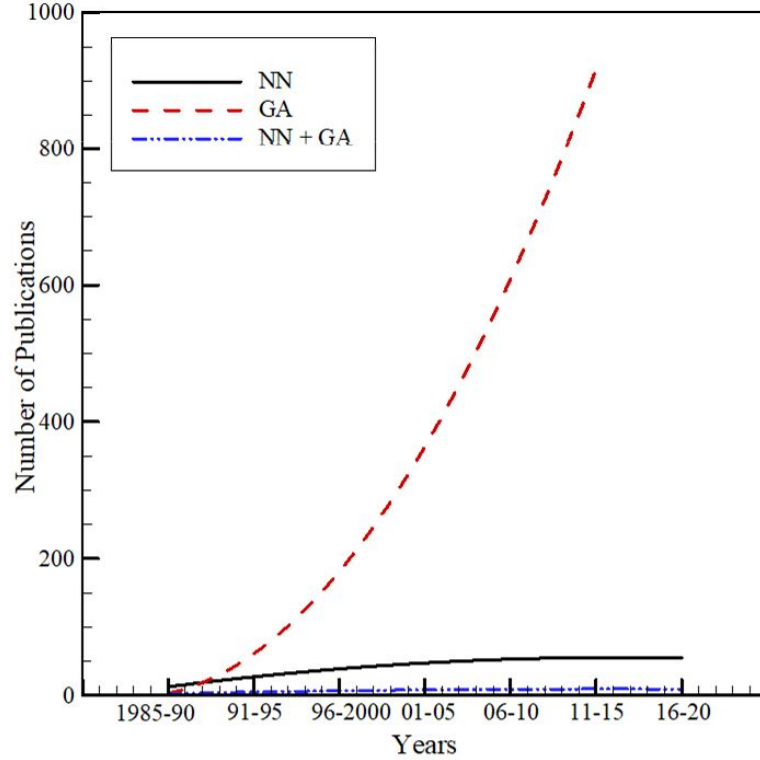


Figure 2.1: Statistical presentation in terms of publication number since 1985 till today on scheduling problems using GA, NN, and hybridization of GA with NN.

2.3.1. NN in Manufacturing Scheduling

Over the last three decades, researchers have been attracted by NN for different areas, especially for manufacturing scheduling. Since the 1980s, NN has been considered for the manufacturing scheduling problems to optimize the completion time (Sabuncuoglu, 1998). The researchers have been attracted to do investigation in the manufacturing scheduling area because of considering the following

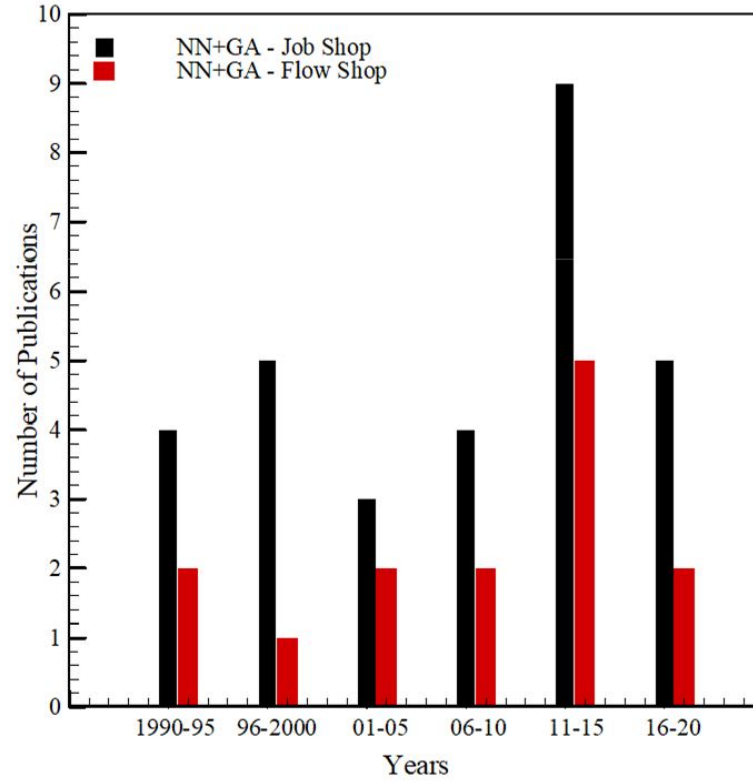


Figure 2.2: Number of publications over last three decades on job shop and flow shop scheduling problems that used GA and NN together.

characteristics of NN ([Akyol and Bayhan, 2007](#)).

- NN can quickly learn the relationship between input and output variables and relate them easily, which is very complex analytically for any other machine learning approaches.
- NN is considered as a very well-performing optimization method.
- NN, e.g., backpropagation NN, can quickly solve the scheduling problem but cannot optimize the problem like Hopfield NN.
- NN is also used for selecting scheduling rules based on the input and output requirement to obtain an accurate estimation between the parameters such as mean flow time, job tardiness, computational time, etc.

- Any static scheduling problem, the traditional methods such as dynamic programming, branch and bound methods can find optimum results. However, These traditional methods are not suitable for flexible and dynamic manufacturing scheduling problems, whereas NN shows better performance for optimizing the scheduling issues.

In 1985, [Hopfield and Tank \(1985\)](#) at first introduced the application of NN for scheduling problems to optimize. Later on, several researchers used their proposed approach for job shop scheduling problems. Following the approach developed by Hopfield and Tank, [Foo and Takefuji \(1988\)](#) tested on 2×3 (i.e., 2 jobs and 3 machines) problem where the computational complexity was $O(m^2n^2 + mn)$. Furthermore, [Lo and Bavarian \(1993\)](#) extended the Hopfield approach and used for 10×3 problem where the computational complexity was $O(\alpha + mnk)^2$, here k defines the processing time. Besides, [Satake et al. \(1994\)](#) modified the Hopfield technique for the 14×7 problem, but still, the computational time was 700 sec, which was not good enough for optimization. [Sabuncuoglu \(1998\)](#) published a review article on NN's application in scheduling problems where the researcher pointed out various limitations of existing approaches and optimization issues.

Starting from 2003 to 2010, Agarwal, with his colleagues, did a comprehensive research on augmented neural networks for production scheduling problem ([Colak and Agarwal, 2005](#); [Agarwal et al., 2006](#); [Kasap and Agarwal, 2012](#)). [Colak and Agarwal \(2005\)](#) researched on one of the meta-heuristic approaches, i.e., augmented NN (Aug-NN) for open-shop scheduling problem. Initially, 3×3 problem (3 jobs and 3 machines) was considered for the problem, and then larger group such as 25×25 , 30×30 , 50×50 , and 100×100 were also considered. In every case, the proposed, designed Aug-NN showed better performance in terms of computational time. In the following year, [Agarwal et al. \(2006\)](#) used Aug-NN for a non-identical machine environment where 4 tasks and 3 machine rules were

considered. The proposed approach for non-greedy rules showed a reduction in the average gap between the lower bound and the solution of 7-10% from regular greedy rules. In 2012, [Kasap and Agarwal \(2012\)](#) investigated the Aug-NN approach for the bin-packing problem, which showed a satisfactory result. 1210 benchmark problems were investigated, whereas 917 problems were solved and optimized its completion time and reduced the gap between upper bound and solution by 0.66%.

Recently, [Huang and Gao \(2020\)](#) investigated a time wave neural network for a time-dependent project scheduling problem where the proposed NN does not need any training. The proposed NN consisted of 7 parts that include input, output, sender, receiver, time window selector, wave generator, and a neuron state. It was found that the designed algorithm performed better while the number of nodes were ranging between 60 and 120 and shows a bad performance when node number less than 30.

2.3.2. GA in Manufacturing Scheduling

GA is used as an optimization algorithm to find the optimal solution with a minimum completion time. It is found that most of the optimization techniques adopt a single object and single solution, whereas GA can handle many solutions from the given population with optimization ([Nguyen *et al.*, 2017](#)). Therefore, GA has been received the most researchers' attention to do study in the manufacturing scheduling field. A short but enlightening survey on the literature of GA starting from 1996 has been presented in this subsection.

Starting from 1996, [Ono *et al.* \(1996\)](#) introduced a job sequence matrix-based GA for job shop scheduling problems. In addition, the researchers also designed a new crossover, which was used to store the features of the system.

Moreover, Giffler and Thompson method was also used, and the proposed approach was used successfully for 10×10 and 20×5 problems. After a while, [Chou \(2009\)](#) used experienced learning GA for a single machine scheduling problem. Exponential smoothing techniques were used to update the job-job matrix and position-job over the generation and build up a relationship between position and jobs. It was found that experienced learning GA could generate 14% chromosome, which was 10% and 12% more than the Lower Bound-Base Bias Roulette (LBBR) and Random process, respectively.

An effective GA was used for flexible job shop scheduling problems by [Zhang et al. \(2011\)](#). The aim of the work was to optimize the makespan of the job. Local Selection was considered with GA in order to generate the initial population. The proposed GA approach took less running time by half compared to the GENACE approach. Furthermore, [Wu et al. \(2011\)](#) also considered the GA technique to minimize the completion time of the job. It was reported that the branch and bound learning technique was used for finding the optimal solution earlier, and then GA was further applied to determine the best optimal solution. The error was recorded less than 0.11% using the GA algorithm for considered scheduling problems.

Besides, [Qing-Dao-Er-Ji and Wang \(2012\)](#) designed a new local search approach to help the GA for job shop scheduling problems. It was noticed that the proposed hybrid GA had an optimal solution with a probability of 1, which was proved and compared with other algorithms. Besides, [Asadzadeh \(2015\)](#) also used a hybrid approach considering GA with agent-based local search, which improved the performance of the GA. The proposed hybrid approach showed better results compared to a single GA algorithm in terms of makespan, where the improvement was recorded as 1.85%. Furthermore, [Mendes \(2013\)](#) used the Monte

Carlo method and single point cross over with GA as a hybrid approach for job shop scheduling problems. The proposed approach could find 50% to 75% best solutions in different instances.

[Wu *et al.* \(2015\)](#) considered GA and branch and bound algorithms for two machine flow shop scheduling problems. It was reported from the experimental results that the proposed GA was effective in finding the optimal solution. Moreover, considering stability and robustness, the GA approach was useful for the flow shop. Besides, [Bhatt and Chauhan \(2016\)](#) prepared a review article on GA based job scheduling problems considering single machine and flexible jobs. Extended GA was proposed for open shop scheduling problems in order to optimize the makespan time by [Rahmani Hosseinabadi *et al.* \(2019\)](#). Mutation and crossover operators were investigated for GA using the 4×4 Taillard Benchmark problem. Different numbers of jobs ranging from 4 to 20 with a number of machines from 4 to 20 were considered for the proposed scheduling problem for extended GA. It was reported that the proposed approach required less computation time compared to other GA and hybrid approaches.

2.3.3. Hybridization of GA

Over the last two decades, in order to design the network more efficiently and in an optimized way to save computational time and effort, hybridization has been considered to combine GA with other algorithms. In this study, the research has been focused on hybridization, which combines GA and NN to fulfill the objective of this thesis. It has been found that the hybridization of GA and NN can be an effective approach to predict and optimize any complex scheduling problems. Therefore, in this section, a comprehensive literature survey has been studied. Fig. 2.3 shows a van diagram to present the possibility of the hybridization option that combines GA with NN and other approaches.

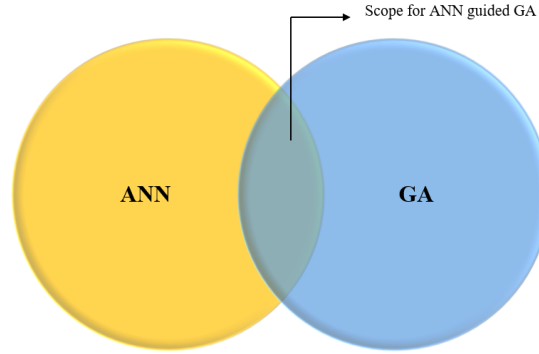


Figure 2.3: Van diagram presenting a possible hybridization option for research.

In 1993, [Dagli and Sittisathanchai \(1995\)](#) proposed a hybrid approach, including GA and ANN, for the discrete environment of a job shop scheduling problem. GA was used to optimize the job schedule where the population was randomly generated, and ANN was used as a multi-criteria evaluator. It was reported, using 10 jobs and 10 machines, the proposed hybrid approach required 291 times to complete the problem, whereas single processing time required 400 times. Later on, [Lee and Dagli \(1997\)](#) also researched on the parallel genetic-neuro scheduler that included six modules (i.e., input, initialization, interpretation, evaluation, population generation, and schedule display) for job shop scheduling problems. It was reported that 50 population with 6 jobs and 6 machines were considered where the proposed approach required only 9 iteration and lead time was 55.

GA with Decision Trees (DTs) as a machine learning technique was proposed for dynamic job shop scheduling problem by [Lee and Dagli \(1997\)](#). The study was conducted considering 1200 jobs with 3 workstations. GA was used to optimize the job dispatch, and a decision tree was used for job release in the job sequence and run concurrently. It was noticed that using ML and GA, the system took 26.05 minutes for 3 workstations and 168.15 minutes for 7 workstations. In addition, [Abe et al. \(2000\)](#) presented a hybrid approach combination of artificial

neural network (ANN) and GA for a 3×3 flow shop scheduling problem. 125 jobs were considered in the investigation. It was noticed that smaller ANN had higher job addition adaptability compared to without ANN.

Yu and Liang (2001) presented NN and GA algorithms for expanded job-shop scheduling problems considering a 6×6 benchmark problem. NN was used to optimize the starting time of the operation for a fixed sequence, and GA was used to find out the optimized sequence. The researchers introduced a new kind of neural network called constrain neural network for their proposed expanded scheduling problem. It was reported that the hybrid approach was remarkably effective for expanded job schedule problems.

Palmes *et al.* (2005) utilized a back mutation-based genetic neural network instead of using backpropagation ANN. The proposed approach made the network dynamic in structure. Moreover, this system helps to consider a wide range of populations with more flexibility and less restriction. Furthermore, Fuqing Zhao *et al.* (2005) also consider a hybrid approach using constraint NN and GA for expanded job shop scheduling problem. NN was used to optimize the job start time, whereas GA was used to optimize the job sequence. 6×6 problem was considered with 55 jobs in the investigation. After compiling the algorithms 100 times, it was noticed that the proposed hybrid approach took 55 run time to achieve the best value, whereas NN and GA needed 71 and 57 times, respectively. Furthermore, Qiang Gao *et al.* (2005) designed a new diffusing operator for global and local search for GA to help feedforward ANN.

Li and Chen (2009) constructed GA and NN as a hybrid approach for dynamic manufacturing scheduling problems. The researcher tried to minimize the

makespan period using the proposed approach. On the one hand, The back-propagation NN (BPNN) was used to demonstrate the arrival of new jobs and machine breakdown status. On the other hand, GA was considered to optimize the sequence of the job and makespan. It was reported that the run time became 18.326 seconds for 920 epochs using the BPNN and GA approach. Furthermore, [Senties *et al.* \(2009\)](#) combined the NN and GA for manufacturing multiobjective scheduling problems. The output of the ANN was sent to the GA for optimizing the job sequence. It was reported that discrete event simulation took 100 times more to complete the job than using ANN with GA, a fast responder for multi-objective scheduling problems.

[Sivapathasekaran *et al.* \(2010\)](#) used GA and ANN in order to maximize the production of the biosurfactant. The statistical experimental strategy was used to obtain the results, which was further used in linking the ANN with GA. It was reported that the proposed hybrid system could boost up the production by nearly 70%. Moreover, [Haq *et al.* \(2010\)](#) also proposed NN and GA as a hybrid approach with random insertion perturbation scheme for the permutation-based flow shop scheduling problem. The NN consisted of 20 and 30 neurons in the two hidden layers. Furthermore, 20 and 50 jobs for 10 machines were considered for scheduling problems.

[Ramanan *et al.* \(2011\)](#) proposed ANN combining with GA for flow shop scheduling problem. ANN was used to obtain the sequence and provided to GA for further improvement and optimize the makespan. The proposed NN was composed of 30 and 20 neurons for 5 jobs and 10 machines. At first, ANN was used to generate the sequence of the jobs for each machine as a solution, and then the output of ANN was sent to GA for optimizing the solution. Besides, [Deane \(2012\)](#) also considered the hybrid approach, i.e., GA and augmented NN,

for scheduling problems used for online advertisement. It was reported that the hybrid approach had the lowest average percentage gap from the upper bound of 1.5% whereas GA and augmented-NN individually had 1.59% and 8.82%, respectively. Moreover, another hybrid option combining GA with perceptron learning rules for machine scheduling was presented by [Fazlollahtabar *et al.* \(2012\)](#). 25 jobs and 5 machines were considered for mathematical modeling. In addition, 150 jobs with 7 machines were also tested, where the learning rate of GA was 0.73.

With his team, Agarwal research the neurogenetic hybrid approach that combines NN and GA for scheduling problems since 2010 ([Agarwal *et al.*, 2010, 2011, 2006](#)). Starting in 2010, [Agarwal *et al.* \(2010\)](#) chose NN and GA algorithms for NP-hard optimization scheduling problem to overcome the difficulties of balancing between global and local search. This proposed approach was used for a population set ranging from 100×5 to 500×30 . The following year, [Agarwal *et al.* \(2011\)](#) proposed a hybrid mechanism, including GA and NN, which is called the Neurogenetic metaheuristic approach for scheduling problems. The researchers used GA and NN for global and local research, respectively. Both GA and NN provided a good solution from the pool irrespectively, and therefore, the overall output of this hybrid approach was better comparatively better than individual GA or NN approach. For example, the proposed Neurogenetic approach had a percentage deviation for 1000 of evaluated schedule was 0.13, whereas GA had 0.19, and NN had 0.25.

A new approach called learning rate optimization genetic algorithm with backpropagation for optimizing NN was considered by [Kanada \(2016\)](#). 50 neurons were used in the hidden layer, and 2 neurons were used in the output layer in the proposed NN system. It was reported that the proposed hybrid system

solved learning rate scheduling and local search control issues. Besides, [Seidgar et al. \(2016\)](#) presented their work combining ANN and nondominated GA for the flow shop scheduling problem. 5 jobs and 2 machines were used for the proposed approach. Besides, [Inthachot et al. \(2016\)](#) considered GA and ANN combinations for estimating the stock price trend in Thailand. In their research, data collected from 2009 to 2014 were used for determining the efficacy of the hybrid system.

[Dorronsoro and Pinel \(2017\)](#) also proposed a hybrid approach combining machine learning, i.e., virtual savant (VS), for independent job scheduling problem. VS was used to generate the initial population that would be adopted for GA to optimize the scheduling. It was reported that a 16×16 problem was used in the square grid with 4 threads where this population was randomly initialized. The hybrid approach had a confidence level of 95% from the statistical study. Furthermore, [Wu et al. \(2018\)](#) combined GA with Particle Swarm Optimization techniques for multimachine scheduling problems and conducted experimental trials as well. The Sum of job processing time-based learning algorithm was used to optimize the tardy jobs. 100 instances were randomly generated for a population size of 20 to 40, where 5000 iterations were proposed by GA. More recently, in 2019, [Azadeh et al. \(2019\)](#) investigated computer simulation, ANN, and GA for flexible flow shop scheduling problems. The output of computer simulation was inserted into ANN, and the output of ANN was provided to GA for reducing the job completion time in the sequence. It was noticed that the proposed hybrid algorithm reduced the error.

2.4. Research Motivation

From Table 2.2 where GA has some advantages over NN and NN has some advantages over GA, which indicates a potential research field to combine these

two algorithms for metaheuristic applications. Besides, it is clearly noticed that limited literature is found on the hybridization of GA with NN for manufacturing scheduling problems. It is reported that GA tries to give the best possible solution and can discard other solutions, which might be an optimal solution. The hybridization of GA and NN has attractive features, such as requiring less training and no premature convergence, which makes it an alternative approach for manufacturing scheduling problems. Therefore, NN, specifically adaptive neural network (ART), can be used for this purpose in order to find out optimal solutions for GA so that GA can explore more regions in order to find out the optimal solution. Interestingly, no literature found where GA has been considered with ART for the flexible flow shop problem. This motivates us to do research in the area of combining GA and ART for discriminating and clustering ordered permutations using neural network and potential applications in ANN Guided Metaheuristics for flexible flow shop scheduling. In this regard, the following chapters will be focused to fulfill the purpose of this thesis to accomplish the motivation.

Chapter 3

Binary Conversion Methods and ART Neural Network

3.1. Introduction

This chapter's main objective is to present two novel techniques for converting ordered permutations to binary vectors for clustering them using Adaptive Resonance Theory (ART) since ART neural network has been used in many applications due to its fast-adaptable learning process and stable operations. In this process, we present a discriminating technique and clustering ordered permutation using ART-1 and Improved-ART-1. The performances of ART-1 and Improved-ART-1 have been investigated, and the proposed binary conversion methods were evaluated under varying parameters and problem sizes. Three performance indicators, i.e., misclassification, cluster homogeneity, and average distance, are considered in the analysis.

This chapter is organized in the following manner: section 3.2 discusses the proposed two types of binary conversion methods with details of considered performance indicators; an introduction of ART, its classification, advantages and

disadvantages, basic architecture, applications of ART are presented in section 3.3; section 3.4 shows a detailed discussion on operational stages, learning algorithm of ART-1 and improved ART-1 techniques; section 3.5 presents numerical analysis and comparison between method 1 (M1) and method 2 (M2) adopting ART-1 and improved ART-1 neural networks and finally this is concluded in section 3.6.

3.2. Proposed Binary Conversion Methods

A binary conversion technique is required for converting ordered permutations to binary vectors. The binary conversion technique can be applied in developing neural network guided metaheuristic algorithm to solve problems where solutions are ordered permutations. Many problems have this attribute. Typical examples include traveling salesman problems, flow shop scheduling, single row facility layout, and many other quadratic assignment problems. Since, in this work, we have considered ART-1 as a neural network learning mechanism for our investigation which requires binary input for clustering; therefore, the ordered permutation has to be converted into equivalent binary. In order to achieve this transformation, two new binary conversion methods that transformed ordered permutation to binary input has been proposed and described.

3.2.1. Method-1

The first proposed binary conversion method is titled Matrix method (M1) because it is formatted in the matrix style. In M1, if we consider a problem size that has ‘N’ number of distinct objects, it will generate a $N \times N$ matrix. Each distinct object of the permutation sequence represents an individual column in that matrix. When an object is considered from this permutation sequence, then the corresponding position of that column will be numbered as ‘1’. A half-width

(HW) parameter is introduced here to decide how many positions to the left and right of the corresponding object number are going to be converted into 1. If the width of the 1s is too wide, all the solutions will be identical. If the width of the 1s is too narrow, then a small change in the permutation may create totally nonidentical matrixes. Therefore, HW should be selected very carefully.

20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
19	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	
17	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	
16	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
15	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	
13	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	
12	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	
11	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
10	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	
9	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	
8	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	
7	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	
6	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	
5	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	
4	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	
1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	
16 11 15 4 7 17 13 12 9 19 8 5 1 10 3 6 14 18 2 20																					
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20																					

Figure 3.1: Method-1 conversion process.

Figure 3.1 presents a 20×20 matrix where a randomly generated permutation sequence of 20 objects has been considered and indicated as: 16, 11, 15, 4, 7, 17,, 20 (bottom 2nd row). The 1st object of the permutation sequence is 16 as shown in Fig. 3.1. So, this 1st object (16) represents a column number (1, column numbers are at the bottom row horizontal position as: 1, 2,.. 20). The corresponding position (row number at the vertical position numbered as: 1, 2,.. 20) of 16 in column number 1 will be numbered as '1', which is colored as "Red". A HW 3 is considered for this example. Therefore, 3 positions to the left and 3

positions to the right of object 16, which is numbered as '1'(colored in "Red"), will be converted into '1'(colored in "Green"). Similarly, all the other objects in the permutation sequence will be converted. After the matrix is created, the input string which will be sent to the ART can be created by concatenating rows of the table into a single row vector.

3.2.2. Method-2

The second binary conversion method which we call Base-2 (M2). The cartesian coordinate is used in M2, where an object's position is considered in a particular sequence. In M2, the cartesian coordinate of the ordered permutation sequence of 'N'objects is determined, which will be a new sequence. This new permutation sequence will then be converted into a base 2 number.

0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	1
1	0	1	0	1	0	0	1	1	1	0	1	0	0	0	0	0	0	1	0
1	0	1	1	1	0	1	0	0	1	0	0	1	0	0	0	1	0	0	1
0	1	1	0	0	0	0	1	0	1	1	0	1	0	1	0	1	1	1	0
1	1	1	0	0	0	1	1	1	0	0	0	1	1	1	1	0	0	0	0
13	19	15	4	12	16	5	11	9	14	2	8	7	17	3	1	6	18	10	20
16	11	15	4	7	17	13	12	9	19	8	5	1	10	3	6	14	18	2	20
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Figure 3.2: Method-2 conversion process.

Figure 3.2 shows an example of M2 conversion technique. Here, the bottom row, i.e., 1, 2, ..., 20, is the actual position of the object, and the middle row represents the randomly taken permutation sequence of 20 objects (16, 11, 15, 4,....., 20; bottom 2nd row). It is seen that the position of object 1 (colored in "Red") is placed at 13 (bottom row numbered as: 1, 2,..., 20) in the table. According to the cartesian coordinate rule, this position 13 of the object 1 will be placed at the 1st position of the row, which is the 1st number of the new sequence. Likewise, all the new positions will be determined to get the new permutation sequence. Then, each object of the new permutation sequences will be converted

into base 2. From this table, the input string, which will be provided as the input of ART-1, can be written by concatenating each binary column's transpose into a single row vector.

3.3. Adaptive Resonance Theory

There are basically two learning mechanism exists in ANN where one is fast learning process and other one is slow process. ART-1 is fast learning process and deals with binary input vectors. In this paper, we have applied our proposed binary conversion M1 and M2 in ART-1 and Improved-ART-1 for numerical analysis purpose.

Adaptive resonance theory (ART) is a neural network learning technique developed by Gali Carpenter and Stephen Grossberg in 1987 ([Carpenter and Grossberg, 1987](#)). Adaptive and resonance of these two words indicate that this technique is suitable for new learning considering the old information. ART system is utilized to clarify various types of brain and cognitive data. The basic ART adopts an unsupervised NN learning mechanism. ART network is well known to solve the stability-plasticity dilemma where stability indicates that the memorizing the learning process, and plasticity defines how flexible ART is to adopt new learning information. ART network generally executes an algorithm in order to cluster the inputs.

3.3.1. Classification of ART

Different types of ART are introduced by researchers for different purposes over time since 1987. The most common ART types are as follows:

- ART-1 ([Carpenter and Grossberg, 1987](#)): ART-1 is the basic and the simplest architecture type of unsupervised ART. ART-1 deals with binary input

vectors during clustering. It changes with the external change to the vigilance parameter. ART-1 architecture consists of two units, i.e., computation unit and supplemental unit.

- ART-2: ART-2 is the extension and upgraded type of ART-1. It can handle continuous or real-valued input vectors during clustering. The main difference between ART-1 and ART-2 is the input layer, where ART-2 has three input layers ([Brito da Silva et al., 2019](#)).
- Fuzzy ART: Fuzzy ART is the expansion of ART and Fuzzy logic ([Carpenter et al., 1991](#)).
- ARTMAP: ARTMAP is a supervised form of ART where it learns from the previous stage, and therefore it is also known as predictive ART ([Carpenter et al., 1991](#)).
- Gaussian ART and ARTMAP ([Williamson, 1996](#))
- TopoART [Tscherepanow \(2010\)](#)
- Hypersphere ART ([Anagnostopoulos and Georgiopoulos, 2000](#))

3.3.2. Advantages and limitation of ART

ART has advantages and limitations, as well. Advantages and disadvantages of ART-1 are as follows ([Brito da Silva et al., 2019](#)):

Advantages of ART:

- i. ART is stable and is not distributed by a wide range of inputs provided to its network.
- ii. It can easily adopt other learning mechanisms to provide excellent and precise results.
- iii. It is better than competitive learning, e.g., BPNN (Back-Propagation Neural

Network), because competitive learning cannot add new clusters when necessary.

iv. ART does not confirm stability while forming clustering.

Limitation of ART:

ART, like ART-1 and Fuzzy ART networks are inconsistent because they depend on the order in which they are trained and on the learning rate.

3.3.3. Applications of ART

Since 1991, ART has been using in different applications due to its fast, efficient, and stable learning technique. ART has been widely used in face recognition, signature verification, mobile robot control, remote sensing, airplane design, autonomous adaptive robot control, target recognition, medical diagnosis, face recognition, land cover classification, fitness evaluation, etc. ([Dagli and Sittisathanchai, 1995](#); [Grossberg, 2013](#); [Burton and Vladimirova, 1997, 1998](#)).

It is noticed that it costs when a new part is introduced during manufacturing. [Smith and Escobedo \(1994\)](#) discussed the importance of ART-1 in engineering applications, especially for information retrieval. ART-1 is forming clusters during the training period using the inputs similar in patterns, and these clusters are saved in the neural database for future reference. when a new input is introduced, ART-1 tries to match this new input with the neural database's stored pattern and process accordingly. [Dagli and Sittisathanchai \(1995\)](#) adapted ART-1 for the machine-part family formation in the cellular manufacturing environment. [Burton and Vladimirova \(1997\)](#) considered ART for fitness evolution for computational optimization using a genetic algorithm. Since NN requires extensive training that consumes enormous time to incorporate with GA; therefore, ART neural network was proposed for GA fitness evolution. However, it was reported that ART generated a higher and uncontrolled number of clusters at high

vigilance value. Later on, [Burton and Vladimirova \(1998\)](#) used ART in the fitness evolution for musical composition. [Hemanth *et al.* \(2010\)](#) used ART to classify brain tumor images and recommended it as an optimal image classifier in the medical field. [Martí *et al.* \(2011\)](#) presented ART based estimation of distribution algorithm (EDA) for multi-object. It was reported that ART based EDA outperformed compared to the multi-objective evolutionary algorithm (MOEA) and multi-objective estimation distribution algorithm (moEDA). [Miguelañez *et al.* \(2004\)](#) considered ART as a classifier in the semiconductor industry for developing an automatic defect identifier. It was found that the proposed ART based system identified 82%.

3.3.4. Architecture of ART

ART is a competitive and self-organizing neural network. ART can be unsupervised (ART1, ART2, etc.) and supervised, such as ARTMAP. The basic architecture of ART includes three units as follows and shown in Fig. 3.3.

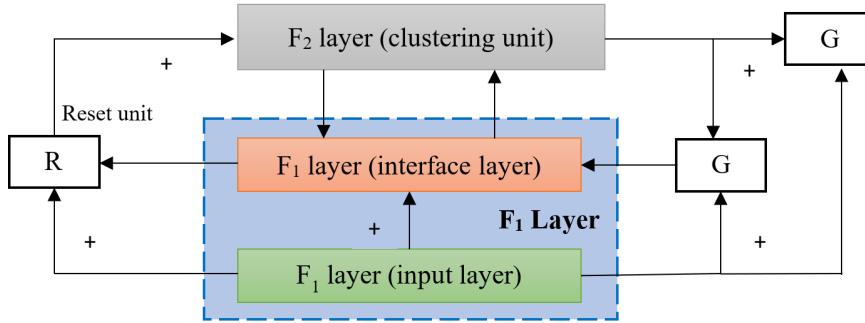


Figure 3.3: Schematic presentation of basic architecture of an ART.

(i) Input/computational unit (F1 layer): F1 layers deal with input vectors and transfer to the next unit F2 layer for clustering purpose. F1 layer includes F1 input and F1 interface layers. F1 input layer works with input vectors only, and there is no processing over there. However, the F1 interface layer works input

vectors according to bottom-up and top-down weights.

(ii) Clustering/recognition unit (F2 layer): This is also known as a competitive layer. F2 layer process all the inputs based on the patterns and then put them into clusters according to the clustering algorithm.

(iii) Control mechanism/reset unit: The final decision is made in the control/reset unit. This reset unit decides whether the cluster unit should learn the input pattern depending on the top-down or bottom-up rules or not. This is generally called the vigilance test, where the vigilance parameter helps to learn new information. There is a supplement unit, which is also known as the gain control unit denoted as “G”.

3.4. ART-1 and Improved ART-1 Learning Mechanism

Many properties make ART-1 more attractive to the researchers ([Smith and Escobedo, 1994](#)). Firstly, as mentioned early, ART-1 is very fast compared to the standard computers because it handles binary inputs. Secondly, ART-1 can easily tackle the implementation of high-performance hardware. Thirdly, ART-1 exhibits stable performance even during new inputs or information. Finally, ART-1 can be described mathematically, which is suitable for design applications. Therefore, ART-1 is a suitable option to consider for manufacturing scheduling problems. Moreover, the manufacturing industries’ outcomes are discrete types; therefore, ART-1 is suitable for the proposed scheduling problem.

3.4.1. ART-1 Learning Technique

Figure 3.4 represents a general architecture of ART-1 that consists of basically two subsystems, i.e., attention and orientating subsystems (Pandya and Macy, 1997). The attention subsystem includes two main layers of ART-1 architecture: the recognition layer and comparison layer with feed-forward and feed-backward features. The attention subsystem responsible for matching the input patterns with the stored patterns and resonance is established if the pattern matches. Furthermore, the orientating subsystem determines the mismatch between the top-down and bottom-up patterns in the recognition layers. Besides, three units in the ART-1 architecture includes Gain-1 and Gain-2 and Reset (as shown in Fig. 3.4).

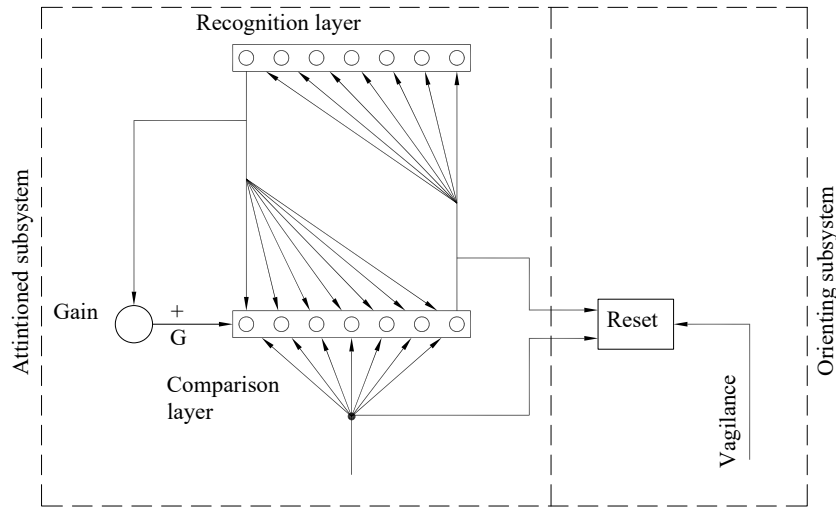


Figure 3.4: Basic architecture of ART-1 (Pandya and Macy, 1997).

Recognition Layer

The recognition layer deals with the input vectors to compare to the original vectors using a factor called vigilance. This vigilance is a measuring parameter

to determine the distance between the input and the fired neuron cluster center in the recognition layer. Here one of the two things has happened, i.e., if the vigilance is below the threshold value, then a new category must be generated, and the input vector must be placed in that new category; otherwise, if the input vector authorizes the vigilance, then the engaging neuron is trained such a way so that the center of the cluster is moved toward to the input. This recognition layer is also termed as F2, which is a top-down layer.

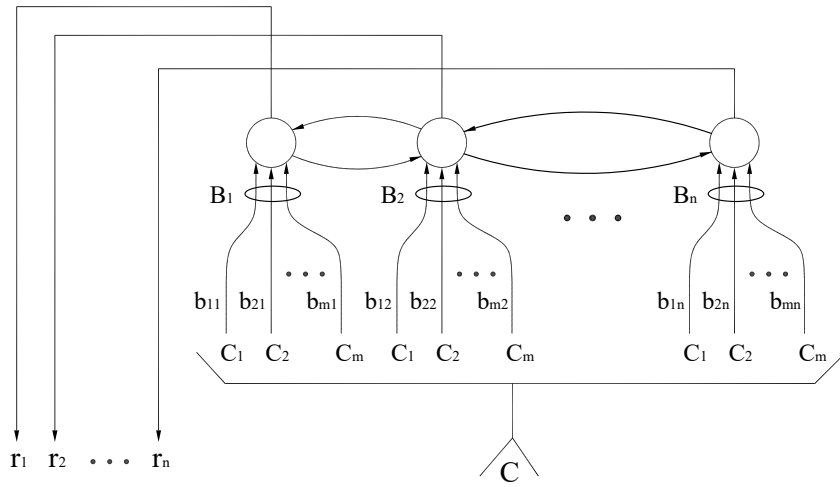


Figure 3.5: Recognition layer of ART-1 (Pandya and Macy, 1997).

Figure 3.5 shows the details of the recognition layer. Each of the recognition layer neuron is denoted as j with a weight vector \mathbf{B}_j . Each neuron receives input from the comparison layer. The net output (ψ_i) of the recognition layer can be written as:

$$\psi_i = \sum_{i=1}^M b_{ij} c_i \quad (3.1)$$

$$r_j = f(\psi_j) = \begin{cases} 1 & \text{for } \psi_j > \psi_i \text{ for all } i \neq j \\ 0 & \text{otherwise} \end{cases} \quad (3.2)$$

Where c_i represents the output of i^{th} neuron in the comparison layer, f is

the step function, r_j is the binary value, M is the total number of neurons in the comparison layer and b_{ij} are the bottom up weights.

Comparison Layer

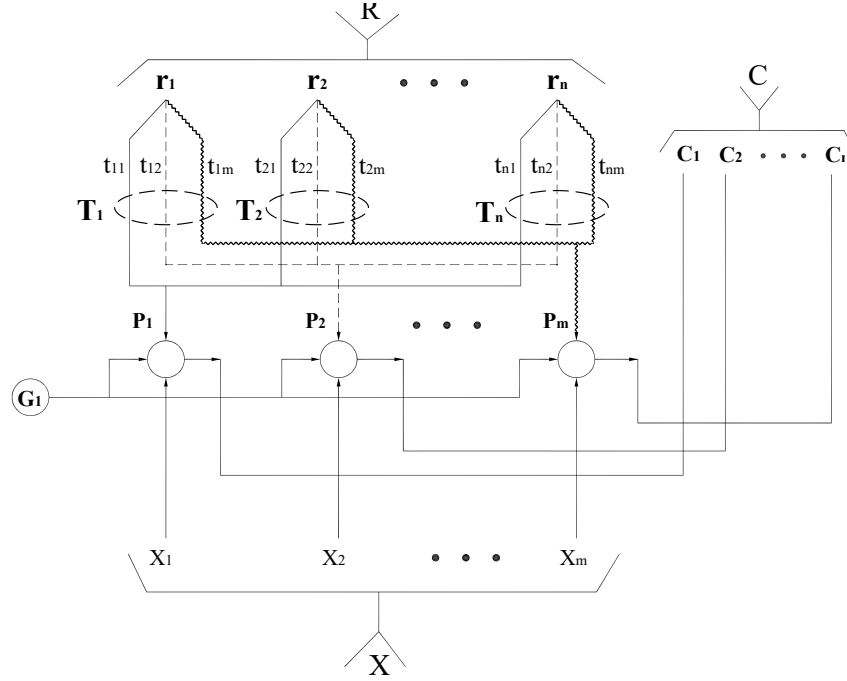


Figure 3.6: Comparison layer of ART-1 (Pandya and Macy, 1997).

The comparison layer is termed as F1 layers, which is a bottom-up layer. A detailed picture is illustrated in Fig. 3.6. The comparison layer deals with three types of inputs such as:

- i) The input pattern X i.e., x_1, x_2, \dots, x_i
- ii) The same gain input to each of the neuron ($G1$).
- iii) The input from recognition layer as the feedback signal. The feedback signal value can be determined by:

$$P_i = \sum_{j=1}^N t_{ji} r_j \text{ for } i = 1, \dots, M \quad (3.3)$$

Where, r_j is the output of the j th neuron of the recognition layer and N is the total number of neurons in the recognition layer. T_j is the weight vector in the comparison layer and t_{ij} are the top down weights. Gain-1 (G-1) is 1 when the Vector \mathbf{R} is 0 and the logical “OR” of the components of the input vectors \mathbf{X} is 1 and given by:

$$G_1 = \left(\overline{r_1 | r_2 | \dots | r_n} \right) \bullet (x_1 | x_2 | \dots | x_M) \quad (3.4)$$

Gain-2 (G-2) is 1 when the logical OR of the component of the \mathbf{X} is 1 and given by:

$$G_2 = (x_1 | x_2 | \dots | x_M) \quad (3.5)$$

The comparison layer adopts a two-third rule which states that if two-third of the inputs are 1 then the output is 1, otherwise the output is 0.

3.4.1.1 ART-1 Learning Stages

There is no input at the initial stage (stage-I), and G-2 becomes 0, according to Eq. (7). When an input is introduced in the network, the input passes through the comparison layer to the recognition layer (see Fig. 3.7). Then the process starts from the recognition layer. The feedback vector \mathbf{R} of the recognition layer is initially set to 0, and according to the Eqs. (6) and (7) G1 and G2 become 1. The output of each neuron in the recognition layer is the dot product of the weight vector \mathbf{B}_j and the output component \mathbf{C} vector of the comparison layer. The winning neuro then triggers other neurons in the recognition layer. Therefore, one of the components of the vector \mathbf{R} becomes 1, and the rest of the components become 0, which starts the comparison phase.

In stage-II, each neuron of the recognition layer is compared to its prototype that is stored in the bottom-up weights with the input pattern and when the best

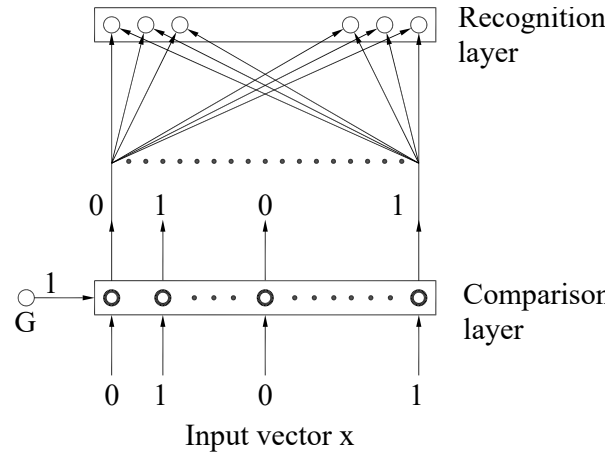


Figure 3.7: Operation stage 1 of ART-1 where G_1 is 1 (Pandya and Macy, 1997).

match is found to be fired (as shown in Fig. 3.8). During this comparison phase, vigilance is set so that the best match is found. Besides, the vector \mathbf{R} becomes 1, whereas G_1 is set 0. According to the two-third rule, neurons with 1's in the comparison layer from \mathbf{X} and \mathbf{P} vector will be triggered.

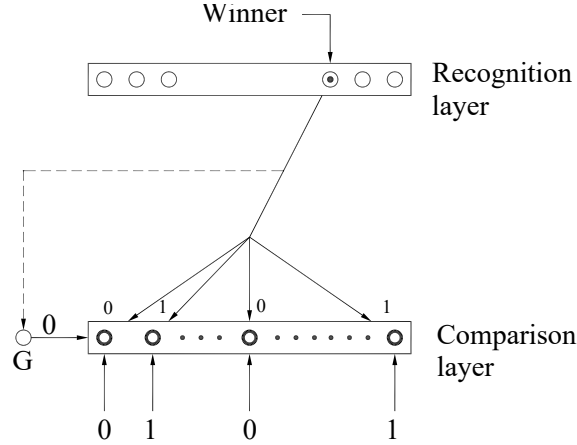


Figure 3.8: Operation stage 2 of ART-1 where G_1 is 0 (Pandya and Macy, 1997).

Let, D , and K are the ones in the \mathbf{X} and \mathbf{C} vector, respectively. Hence, \mathbf{S} is the similarity ratio which can be written as $\mathbf{S} = K/D$. This similarity vector \mathbf{S} , is a matrix that will be used to determine likeness between the input vector and the prototype. Now, a criterion will be set based on which the cluster will be

accepted and rejected. The criterion will be given by:

$$\mathbf{S} > \rho \longrightarrow \text{passed the vigilance test}$$

$$\mathbf{S} \leq \rho \longrightarrow \text{passed the vigilance test}$$

If the vigilance is passed, it means there is no difference between the input vector and the prototype, and therefore, the required action is stored in the center of the winning neuron cluster. Besides, there is no reset value in this case, and the network operation is completed. Now, if the vigilance is not passed, it means the value of \mathbf{S} is below than the threshold; then the input patten is placed into a new cluster center of the neuron by creating a new category rather than putting into the winning neuron, which will be inhibited and this is done by the reset signal. This the end of the comparison phase (see Fig. 3.9).

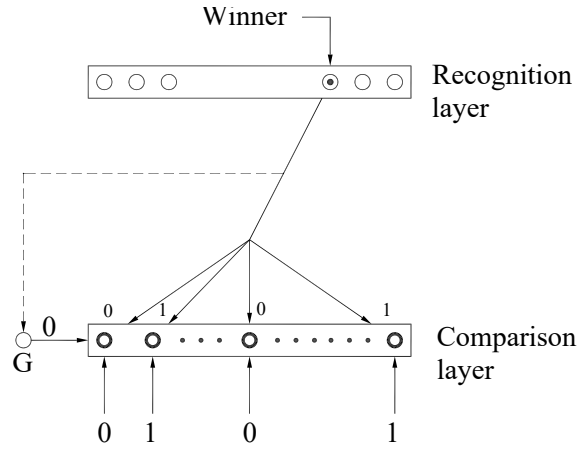


Figure 3.9: Operation stage 3 of ART-1 (Pandya and Macy, 1997).

In the fourth stage (as shown in Fig. 3.10), the search phase is initiated if there is no reset signal generated, and the match is considered satisfactory. At this point, classification is complete. If it does not happen, then the vector \mathbf{R} is set to 0, and the $G1$ becomes 1 again so that the input vector \mathbf{X} appears on \mathbf{C} and a new neuron is triggered for the recognition layer to be won, and this

process will be repeated until the winner neuron that passes the vigilance test ($S > \rho$).

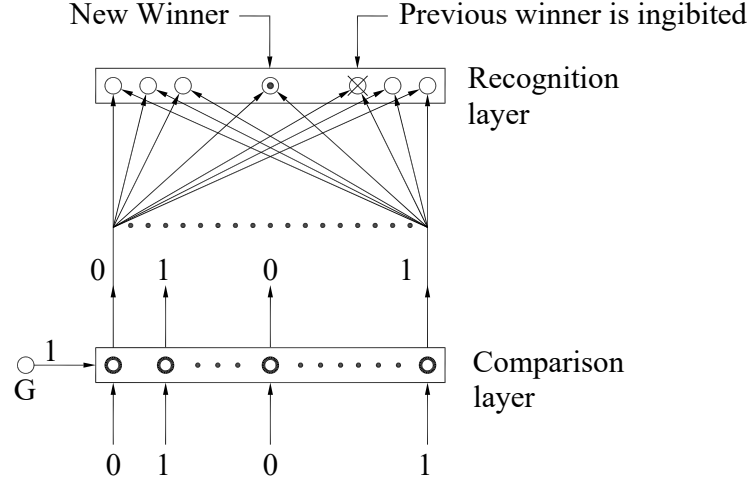


Figure 3.10: Operation stage 4 of ART-1 (Pandya and Macy, 1997).

3.4.1.2 ART-1 Learning Algorithm

The algorithms of ART-1 is as follows (Pandya and Macy, 1997) :

Step 1: Initialize the vigilance parameter, learning rate and weights which is as follows:

$$b_{ij} < \frac{L}{L - 1 + x} \quad (3.6)$$

where, x is the number of input vector, L is a constant.

Step 2: When an input vector is introduced in the network then the recognition layer starts comparison and find out the maximum of all the net output of the neurons according to Eq. 3.1.

Step 3: Run the vigilance test. A neuron (j) in the recognition layer passes the vigilance test if only if

$$\frac{\psi_j}{\sum_{i=1}^N x_i} > \rho \quad (3.7)$$

where ρ is the vigilance threshold.

Step 4: If the vigilance test fails then obscure the current winner and go to the step 1 to figure out another winning neuron. Repeat the whole process until a winning neuron passes the vigilance test, and then go to step 5.

Step 5: If there is no neuron that passes the vigilance test then create a new neuron in order to accommodate the new input pattern.

Step 6: Adjust and update the feed-forward weights from the winning neuron to the inputs. The update of the bottom-up and to-down weights can be done as follows:

$$b_{ij} < \frac{Lc_1}{L - 1 + \sum c_k} \quad (3.8)$$

$$t_{ij} = c_i \quad (3.9)$$

Step 7: If there is no input vector then stop otherwise go to Step 2.

3.4.2. Improved-ART-1 Learning Technique

The general form of ART-1 does not provide satisfactory outcomes due to: (i) the classification process and the outcomes are dependent on the order of the applied input vectors; (ii) the stored patterns become scattered when more inputs are inserted which means that as the training proceeds the stored vectors become more spares and spares; (iii) it is difficult to identify the suitable vigilance parameters since cell number increase with higher vigilance value. These drawbacks make ART-1 not suitable for manufacturing scheduling problems, especially for permutation-based flow shop problems. As a result, [Dagli and Huggahalli \(1993\)](#) proposed improved ART-1 by changing a few things in the basic methods. The changes are as follows:

- (i) Colum and rows will be arranged according to the decreasing number of 1s.
- (ii) The prototype patterns are stored during training period according to one of

the following equations:

If the number of 1s in the weight vector \mathbf{T} is larger than input vector \mathbf{X} then Eq. 3.6 becomes:

$$b_{ij} < \frac{Lt_{ij*}}{L - 1 + \sum t_{ij*}} \quad (3.10)$$

where, $t_{ij*} = t_{ij*}$ (no change).

If the number of 1s in input vector \mathbf{X} is larger than the weight vector \mathbf{T} then Eq. 3.6 becomes:

$$b_{ij} < \frac{Lx_i}{L - 1 + \sum x_i} \quad (3.11)$$

3.5. Numerical Study and Performance Analysis

In this section, the proposed binary conversion methods have been considered for numerical studies. Moreover, an experimental data generation process has been described, which is used during computational programming. Besides, a comparative study has been presented between the binary conversion methods M1 and M2, considering ART-1 and Improved-ART-1.

3.5.1. Experimental Data Generation

Experimental data is generated in order to analyze the presented binary conversion techniques and neural networks. At first, certain numbers of permutations are randomly generated which are called seeds. Then, each seed is perturbed independently, which is termed as degree of perturbation, to produce other sequence of permutations and it is indicated as the number of solutions per seed. These permutations are generated in random order. It can be predicted from these experimental data ahead of time that how many clusters are supposed to

be generated. Therefore, if the number of seeds is 'X', then potentially the number of generated cluster should be also 'X'. Each permutation of a particular cluster is called member of that cluster. Here, three indicators have been introduced as performance evaluator for the numerical work which are as follows:

3.5.1.1 Misclassification

It is important to determine misclassification whether a given permutation classified correctly or not. If the distance between a particular permutation and a given cluster centroid is greater than the distance between that permutation and any other cluster centroid to where the permutation does not belong, then that permutation is termed as misclassified. Therefore, misclassification is one of the important performance indicators in this analysis. Lets say, cluster i ($i = 1, 2, \dots, I$ where I is number of clusters) contains total number of P_i permutations where each permutation is indicated by $p = 1, 2, 3, \dots, P_i$. The cartesian coordinate of permutation p can be defined as, $\Pi_p = \{O_{1,p}, O_{2,p}, \dots, O_{N,p}\}$; where, $O_{n,p}$ is the location of object- n in permutation p , where $n = 1, 2, \dots, N$, and N is the number of distinct objects. The centroid of cluster- i defined as, $C_i = \{C_{i,1}, C_{i,2}, \dots, C_{i,N}\}$ where the n^{th} cartesian coordinate is given as:

$$C_{i,n} = \frac{\sum_{p=1}^{P_i} O_{n,p}}{P_i}; \forall n \quad (3.12)$$

The distance D_{p,C_i} of permutations p from the cluster centroid C_i to which it belongs can be calculated by:

$$D_{p,C_i} = \sqrt{\sum_{n=1}^N (C_{i,n} - O_{n,p})^2} \quad (3.13)$$

The $D_{p,C_{i'}}$ of permutation p to any other cluster (cluster- i') to which the permutation does not belong can be calculated in a similar fashion. If $D_{p,C_i} > D_{p,C_{i'}}$

for at least one $i' \neq i$, the permeation is said to be misclassified.

3.5.1.2 Homogeneity

Homogeneity is defined as the largest count of members from the same seed divided by the total number of permutations that belong to that cluster. If most of the members are from the same seed, then the homogeneity of that cluster becomes large, and that is considered a better cluster. Generally, cluster homogeneity is calculated in percentage (%). Hence, a high percentage of cluster homogeneity is expected in the analysis.

3.5.1.3 Average Distance

Average distance is another indicator that can be considered to evaluate the performance comparison between binary conversion methods M1 and M2. The average distance can be calculated from each member's total distance from the centroid of a cluster divided by the total number of members of that cluster. It is noted that if the average distance is less, then the cluster formation is good. Therefore, a lower average distance is presumed to indicate a good clustering criterion.

In the following subsections, the comparison between two binary conversion techniques has been discussed in terms of misclassification, homogeneity, average distance. The performance and behavior of ART-1 and Improved-ART-1 have been investigated while clustering these binary data considering different vigilance values, the number of seeds, half-width, degree of perturbations, problem size.

3.5.2. Comparison between M1 and M2 with ART-1

At first, the performance of the proposed binary conversion methods has been examined for ART-1. In this initial analysis, the number of seed 5, degree of

perturbation 0.4, vigilance value 0.5, number of solution per seed 500, half-width 3, and number of objects 40 have been considered to perform the numerical study. Since the proposed analysis provides slightly different solutions during each run; therefore, a total of 10 solutions have been considered and taken the average of these 10 solutions. As a result, ART-1 considering M1, generates a total number of 9 clusters where the total number of misclassification is 1834 (see Table 3.1). Similarly, Table 3.2 presents the performance of ART-1 using proposed binary conversion M2, where the number of clusters is 9, same as in ART-1 with M1, but the number of misclassification for ART-1 M2 is 1900 that is higher than ART-1 M1. Hence, binary conversion M1 overall outperforms M2 for ART-1.

Table 3.1: Performance Analysis for ART-1 considering M1.

No. of Cluster	Misclassification	Cluster ID	No. of Members	Homogeneity (%)	Average Dist.
9	1834	0	308	50	55
		1	335	37	62
		2	281	44	59
		3	342	48	55
		4	382	49	57
		5	290	42	58
		6	258	44	59
		7	184	56	33
		8	108	26	29
		9	58	20	4

Table 3.2: Performance Analysis for ART-1 considering M2.

No. of Cluster	Misclassification	Cluster ID	No. of Members	Homogeneity (%)	Average Dist.
9	1900	0	356	34	64
		1	317	33	65
		2	321	37	63
		3	331	32	65
		4	365	36	63
		5	366	42	59
		6	216	40	60
		7	165	39	37
		8	148	16	54
		9	59	10	20

A comparison between M1 and M2 adopted ART-1 has been illustrated in Fig. 3.11 in terms of homogeneity for each cluster based on Table 3.1 and 3.2. It is noticed that the clusters made using M1 are more homogeneous than M2 for ART-1. The higher the homogeneity, the better the cluster formation. Besides, according to this bar graph, M1 shows better performance based on homogeneity value than M2 deploying ART-1 (see Fig. 3.11).

Figure 3.12 presents a bar graph where a comparison between M1 and M2 has been presented in terms of average distance at each cluster for ART-1. It is found that the average distance in each cluster for M1 is less than M2, which indicates that the members of a particular cluster are closer to each other. If the average distance is less, the cluster formation is better. According to this result, M1 performs better than M2, considering the average distance and different cluster IDs using ART-1.

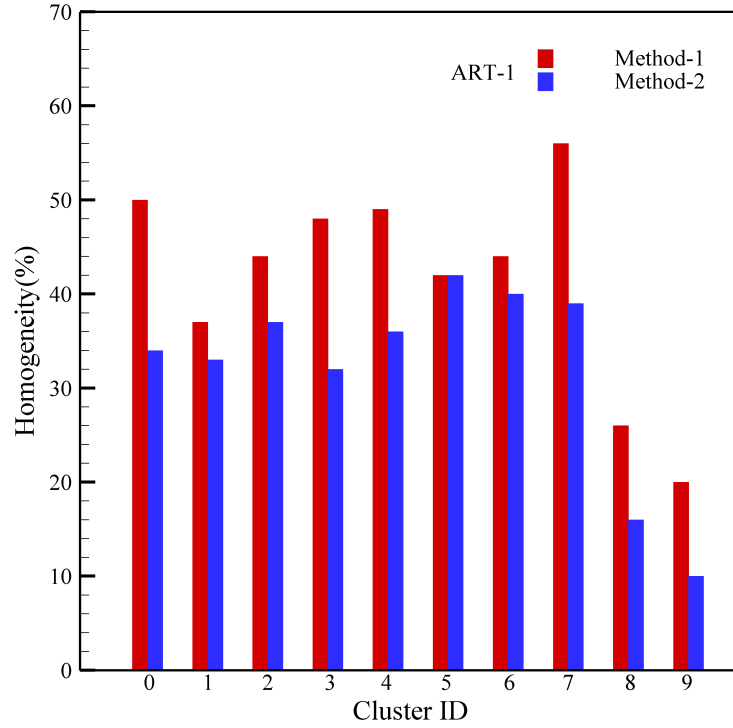


Figure 3.11: Change in homogeneity along with cluster ID for M1 and M2 considering ART-1.

3.5.3. Comparison between M1 and M2 with Improved-ART-1

Similar performance analysis has been conducted to compare M1 and M2, considering Improved-ART-1 in terms of homogeneity for each cluster. Likewise, the number of seed 5, degree of perturbation 4, vigilance value 0.5, number of solution per seed 500, half-width 3, and number of objects 40 have been considered to perform the numerical study for Improved-ART-1. The results show that Improved-ART-1 considering M1 generates a total number of 5 clusters where the total number of misclassification is 22 (see Table 3.3). However, considering binary conversion method-2 for Improved-ART-1, the number of clusters is 5, which is the same the Improved-ART-1 M1, but the number of misclassification for Improved-ART-1 M2 is 183 that is higher than Improved-ART-1 adopting

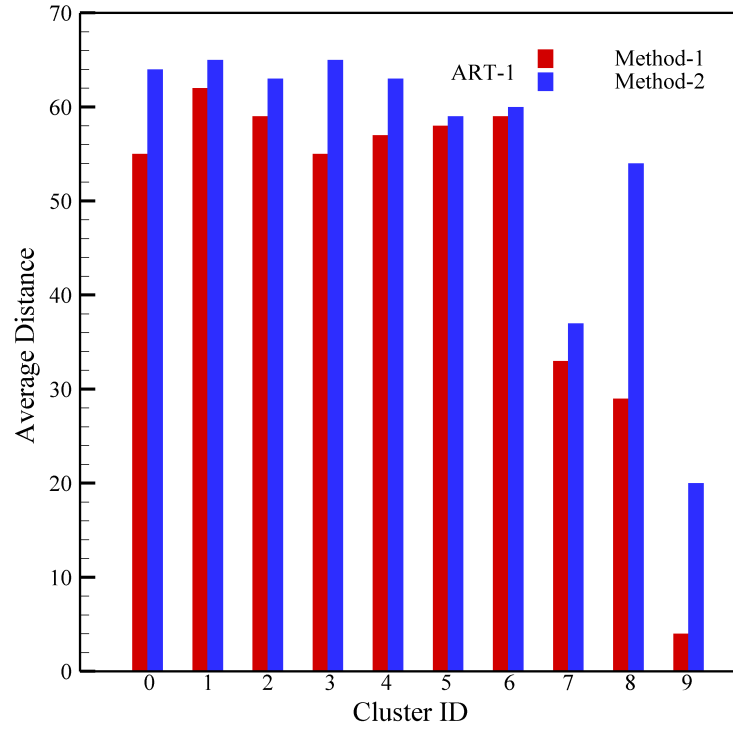


Figure 3.12: Average distance vs cluster ID for M1 and M2 considering ART-1.

M1.

Table 3.3: Performance Analysis for Improved-ART-1 considering M1

No. of Cluster	Misclassification	Cluster ID	No. of Members	Homogeneity (%)	Average Dist.
5	22	0	505	99	21
		1	508	98	22
		2	505	99	21
		3	503	99	21
		4	479	100	20

Table 3.4: Performance Analysis for Improved-ART-1 considering M2

No. of Cluster	Misclassification	Cluster ID	No. of Members	Homogeneity (%)	Average Dist.
5	183	0	697	76	39
		1	602	86	32
		2	613	84	36
		3	500	95	26
		4	88	20	21

A comparison between M1 and M2 using Improved-ART-1 has been presented in terms of homogeneity of each cluster (see Fig. 3.13) considering Table 3.3 and 3.4. It is noticed from the tables that M1 is more homogeneous than M2 for each cluster considering Improved-ART-1. Figure 3.14 presents a comparison between M1 and M2 in terms of average distance at each cluster. It is found that the average distance in each cluster of Improved-ART-1 M1 is less than M2, which indicates that the members of a particular cluster are closer to each other. Therefore, M1 is also performing better than M2 for Improved-ART-1.

A comparison has been made between ART-1 and Improved-ART-1 considering misclassification in terms of proposed binary conversion M1 and M2. Figure 3.15 represents a pie chart that shows the number of misclassifications that have been generated by ART-1 M1, ART-1 M2, Improved-ART-1 M1, and Improved-ART-1 M2. ART-1 using M1 provides a smaller number of misclassifications than ART-1 with M2. Moreover, Improved-ART-1 with M1 has fewer misclassifications than Improved-ART-1 adopting M2, even this much less than ART-1 using M1. Therefore, binary conversion M1 has been chosen for Improved-ART-1 and ART-1 for further analysis based on the various degree of perturbation, vigilance value, and half widths.

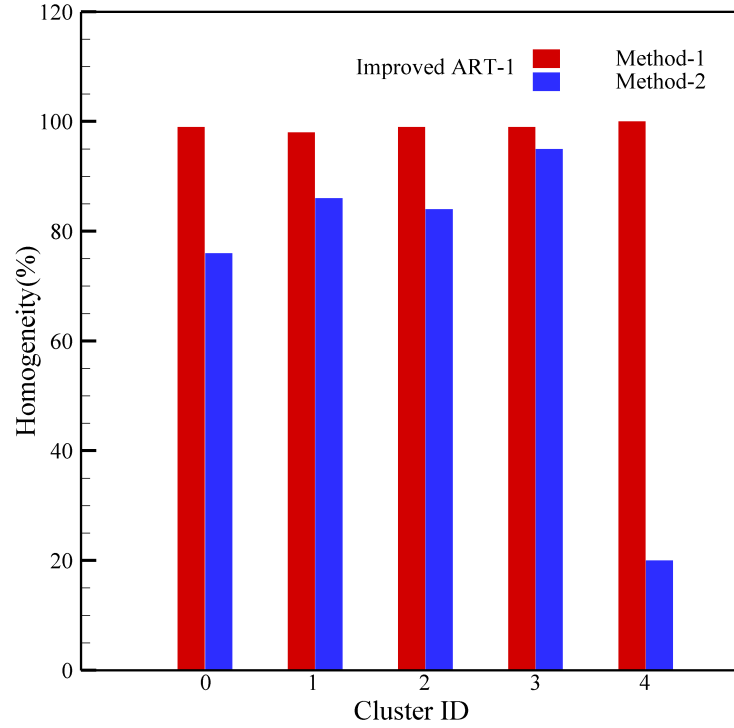


Figure 3.13: Change in homogeneity along with cluster ID for M-1 and M-2 of Improved-ART-1.

3.5.4. Comparison between ART-1 and Improved-ART-1 with M1

In this analysis, a comparison has been illustrated between two best techniques considering ART-1 and Improved-ART-1 using binary conversion M1. Figure 3.16 shows the comparison in terms of misclassification, homogeneity, and average distance. It is seen that the number of misclassifications of ART-1 is nearly 99 % higher than Improved ART-1. Additionally, when homogeneity and average distance are considered performance evaluators, improved-ART-1 still shows better results than ART-1. Therefore, Improved-ART-1 with M1 shows good quality solutions.

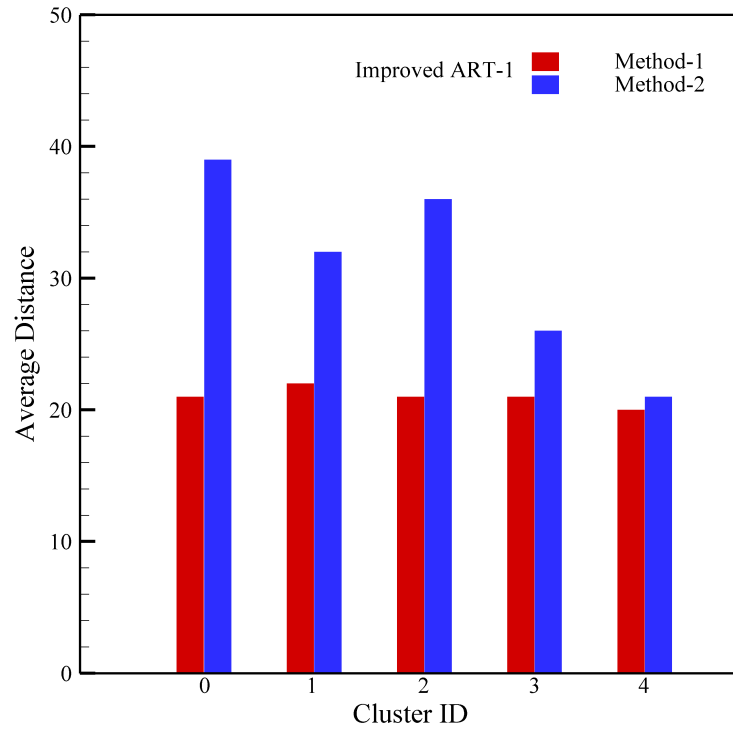


Figure 3.14: Change in average distance with cluster ID for M-1 and M-2 of Improved-ART-1.

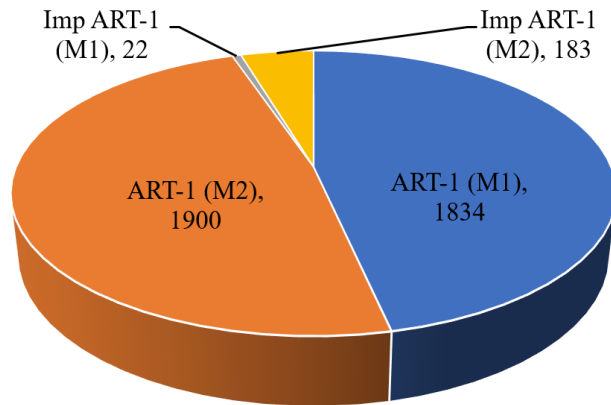


Figure 3.15: Change in misclassification between two methods for ART1 and Improved-ART-1.

A projection between the expected cluster numbers and actual cluster generated by ART-1 and Improved-ART-1 using conversion method-1 has been illustrated in Fig. 3.17. It is reported that the Improved-ART-1 (M1) fits and

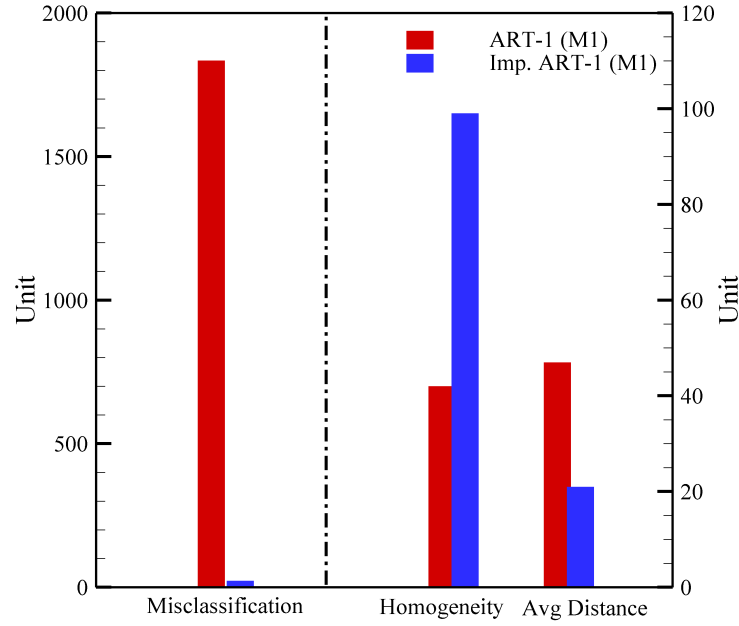


Figure 3.16: Comparison between ART-1 (M-1) and Improved-ART-1 (M-1).

follows the expected cluster number with an increasing number of seeds until 6 at vigilance value of 0.5, perturbation 4, and half-width 3, whereas the number of solution per seed is 500. In contrast, ART-1 does not fit with the expected cluster number, where it generates a higher cluster number than expected. Therefore, this analysis is another good indicator that Improved-ART-1 (M1) can provide a better solution than ART-1.

It is clearly noticed from the previous analysis that the Improved-ART-1 provide good solutions in terms of binary conversion method M1. Now, the investigation has been extended for different problem sizes (i.e., 10 objects as problem-1 (P1), 20 objects (P2), objects 30 (P3), and objects 40 (P4) keeping the parameters that have been mentioned in the earlier analysis unchanged for ART-1 (M1) and Improved-ART-1 (M2). Table 3.5 shows Improved-ART-1 using

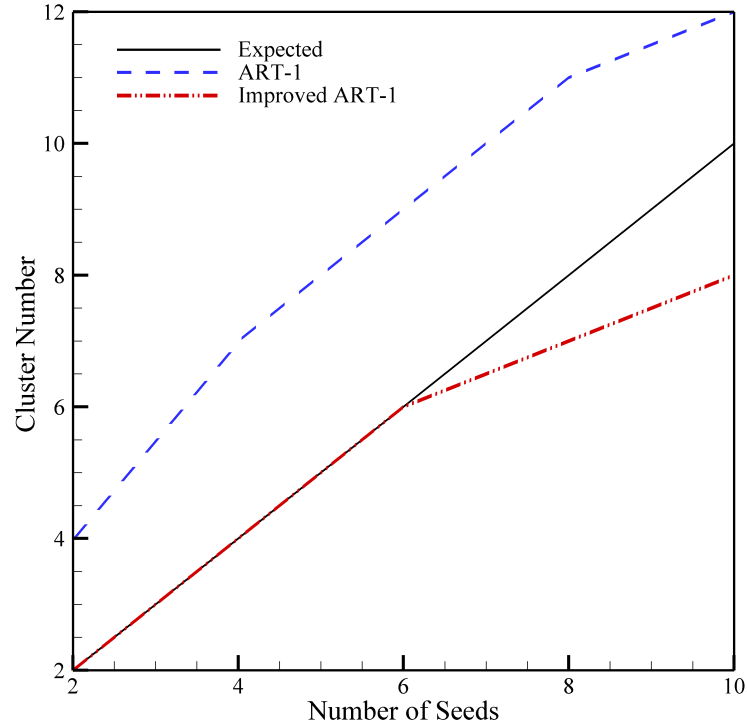


Figure 3.17: Comparison between ART-1 and Improved-ART-1 with M1 at different number of seeds at vigilance value 0.5.

M1 provides a good solution for every problem size. Less misclassification, higher homogeneity, less average distance, and exact cluster number which matches with no of seeds have been achieved.

3.5.5. Effect of Half-Width on Performance for Improved-ART-1 using M1

The comparative study has been further extended by examining the changing effect of half-width considering four problem sizes for Improved-ART-1 with M1 to determine a half-width range where ART will provide the best results. All the other parameters are kept the same for this analysis. Here, the expected number of clusters is 5 as the number of seeds for this problem is 5. It is noticed from Table 3.6 that only one cluster is generated when the half-width is more than

Table 3.5: Comparison between ART-1 and Improved-ART-1 considering different problem sizes. here, C= no. of cluster, M = misclassification, H= homogeneity, Mi= minimum, Mx= maximum, A= avergae, D= distance

	P1 (10 objects)						P2 (20 objects)						P3 (30 objects)						P4 (40 objects)					
	ART-1		Imp. ART-1		ART-1		ART-1		Imp. ART-1		ART-1		ART-1		Imp. ART-1		ART-1		ART-1		Imp. ART-1		ART-1	
	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2
C	9	13	6	8	9	9	8	9	5	5	8	8	9	5	5	9	9	9	5	5	5	5	5	5
M	2059	2215	941	1407	1829	1984	1921	1921	58	58	170	170	1726	46	46	176	1834	1900	22	22	183	183	183	183
H	26	26	39	52	44	34	34	34	96	96	85	85	47	98	98	75	42	35	99	99	72	72	72	72
M. D.	6	6	3	4	13	16	26	26	4	4	5	5	24	5	5	9	39	42	5	5	14	14	14	14
Mx. D.	10	10	8	11	28	28	46	46	32	32	32	32	52	59	59	52	77	79	80	80	80	80	80	80
A. D.	8	8	5	7	18	20	33	33	11	11	12	12	33	17	17	18	47	55	21	21	8	8	8	8

5 for the 1st problem size (10 objects), which is less than the expected cluster number (5). It means that with the increment of the half-width size, all solutions are becoming identical and must be discarded. Hence, the misclassification is also decreasing with increasing half-width. Therefore, it can be stated for problem 1 that the optimal range of half-width is between 1 ~ 4. Likewise, for problem 2, which consists of 20 objects, problem 3 with 30 objects, and problem 4 of 40 objects, the range for half-width is 1 ~ 4, 1 ~ 5, and 0 ~ 5, respectively. In addition, it is noticed that half-width is proportional to the problem size. It is vital to take the optimal half-width for getting quality solutions. Furthermore, this analysis also validated that the half-width that is randomly selected (3) for 40 objects at the very beginning is within the range, which is giving us the best solutions.

3.5.6. Effect of Degree of Perturbation on Improved ART-1 using M1

In this analysis, the effect of changing the degree of perturbation on Improved-ART-1, considering M1 for 40 objects, has been studied. Other parameters such as vigilance value 0.5, half-width 3, number of seeds 5, solution per seed 500 are considered. Figure 3.18 shows the behavior change of Improved-ART-1 (M1) by varying the degree of perturbation in terms of misclassification. It can be stated that the misclassification number is increasing with the increment of the degree of perturbation. Additionally, this graph indicates that misclassification with respect to the degree of perturbation is a proportional performance evaluator. Moreover, there is a big jump in the misclassification when the degree of perturbation increases beyond 10. As the lowest misclassification is expected for the solutions; therefore, the degree of perturbation must be minimal in order to obtain the best solution from ART-1.

Table 3.6: Performance analysis for Improved-ART-1 with binary conversion M1 considering different problem sizes at various half widths.

	P1 (10 objects)					P2 (20 objects)					P3 (30 objects)					P4 (40 objects)				
Half width	0	5	10	15	20	0	5	10	15	20	0	5	10	15	20	0	5	10	15	20
No of cluster	18	4	1	1	1	10	4	3	3	3	1	6	5	3	2	2	5	5	4	3
Misclassification	1895	558	0	0	0	1075	252	188	218	0	256	88	242	207	258	1	26	87	125	166

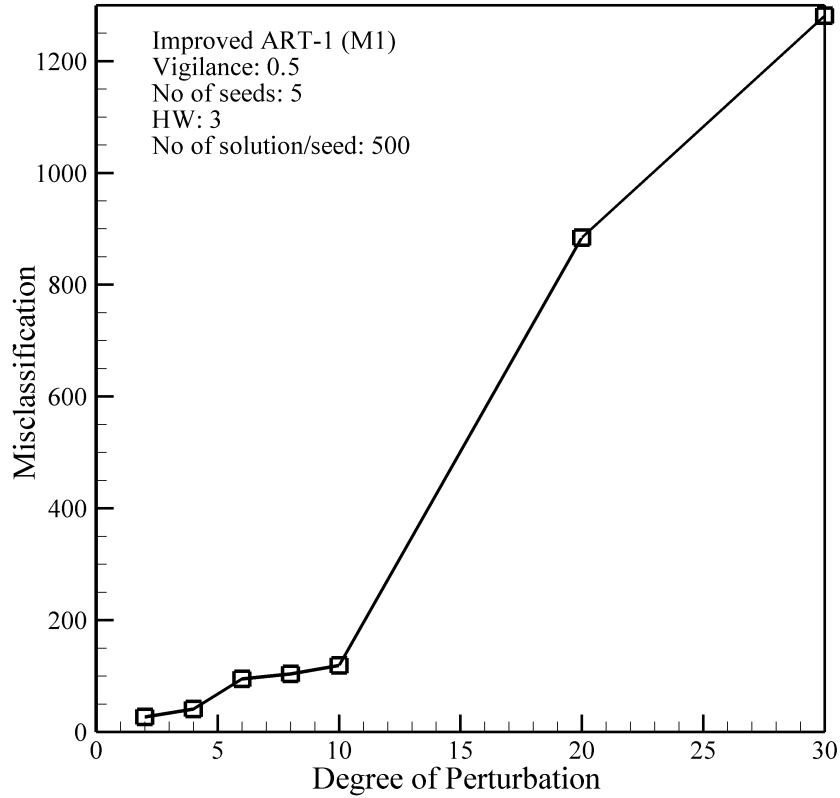


Figure 3.18: change in misclassification as a function of degree of perturbation for Improved-ART-1 using conversion methods M1.

Figure 3.19 represents the sensitivity in terms of incremental rate in percentage of homogeneity and average distance with the degree of perturbation for Improved-ART-1 (M1). The incremental rate of both homogeneity and average distance increases with an increasing degree of perturbation. However, it is evident that the changing rate of homogeneity is very slow compared to the degree of perturbation. It is worth mentioning from the analysis that homogeneity is less sensitive than average distance on the degree of perturbation. The change in average distance is higher than the homogeneity with the degree of perturbation, which also indicates that homogeneity is a good indicator to evaluate the performance of ART-1, which is less dependent on the degree of perturbation.

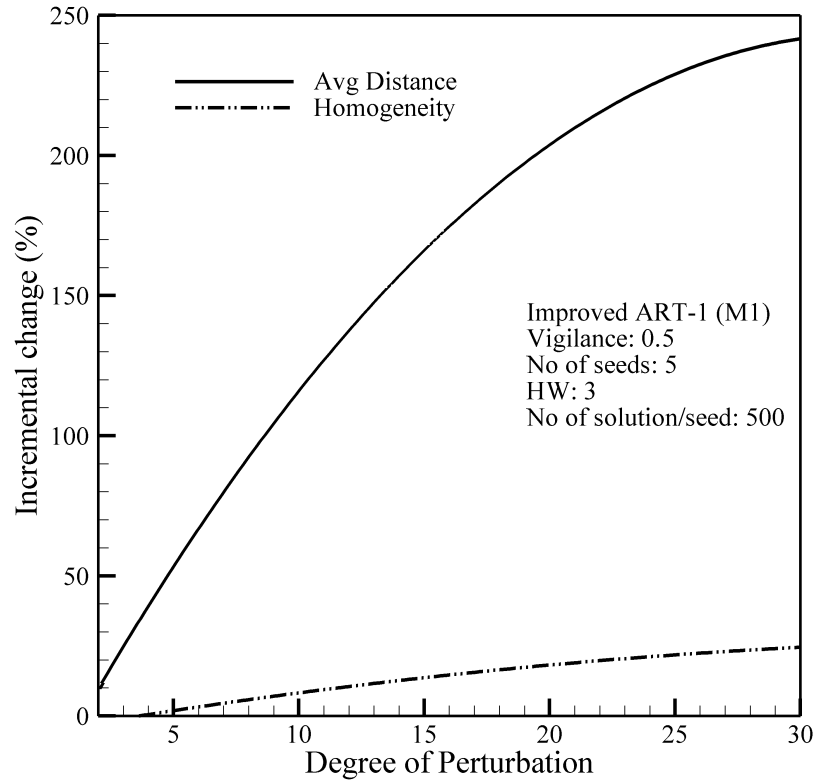


Figure 3.19: Change Homogeneity and average distance with degree of perturbation for Improved-ART-1

3.5.7. Effect of Vigilance Value on Improved-ART-1 with M1

Table 3.7 shows the details of performance analysis of varying vigilance parameter on Improved-ART-1 for 40 objects and keeping other parameters the same as fixed before. It is noticed that number of clusters, homogeneity, and average distance are almost identical for vigilance value ranging from 0.1 to 0.7. Therefore, it can be said that Improved-ART-1 (M1) is less sensitive on vigilance value, which is a very good indicator of choosing Improved-ART-1 over ART-1.

Table 3.7: Performance analysis for Improved-ART-1 (M1) considering 40 objects for different vigilance value.

	Vigilance value							
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
No of cluster	4	5	5	5	5	5	5	9
No of member	500	490	500	500	500	500	500	250
Homogeneity	70	80	99	99	99	99	99	49
Min Distance	14	11	5	5	5	5	5	35
Max Distance	73	78	80	80	80	80	84	78
Avg Distance	26	24	21	22	21	21	21	49

Since vigilance value has less effect on Improved-ART-1; therefore, the same analysis has been performed to determine the dependency of vigilance value on ART-1 (M1) (see Table 3.8). Table 3.8 shows that other than the number of clusters, homogeneity, and average distance vary with different vigilance values. It indicates that ART-1 (M1) highly sensitive and dependent on vigilance value.

Table 3.8: Performance analysis for ART-1 (M1) considering 40 objects for different vigilance value.

	Vigilance value							
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
No of cluster	8	9	9	9	9	9	9	9
No of member	278	250	278	271	250	250	250	250
Homogeneity	46	50	107	58	42	45	47	49
Min Distance	33	32	41	33	38	36	38	35
Max Distance	72	73	80	81	73	72	76	78
Avg Distance	110	44	54	46	49	48	50	49

Here, another most important factor, i.e., misclassification, has been considered to verify ART's dependency on vigilance value. Figure 3.20 illustrates a comparison between ART-1 and Improved-ART-1, where misclassification is changing with vigilance value. As it is noticed that ART-1 is varying drastically with increasing vigilance value, whereas Improved ART-1 is behaving relatively stable between 0.3 to 0.7 for 40 objects, half-width 3, number of seeds 5, degree of perturbation 4. Therefore, Improved-ART-1 is less sensitive to vigilance value than ART-1.

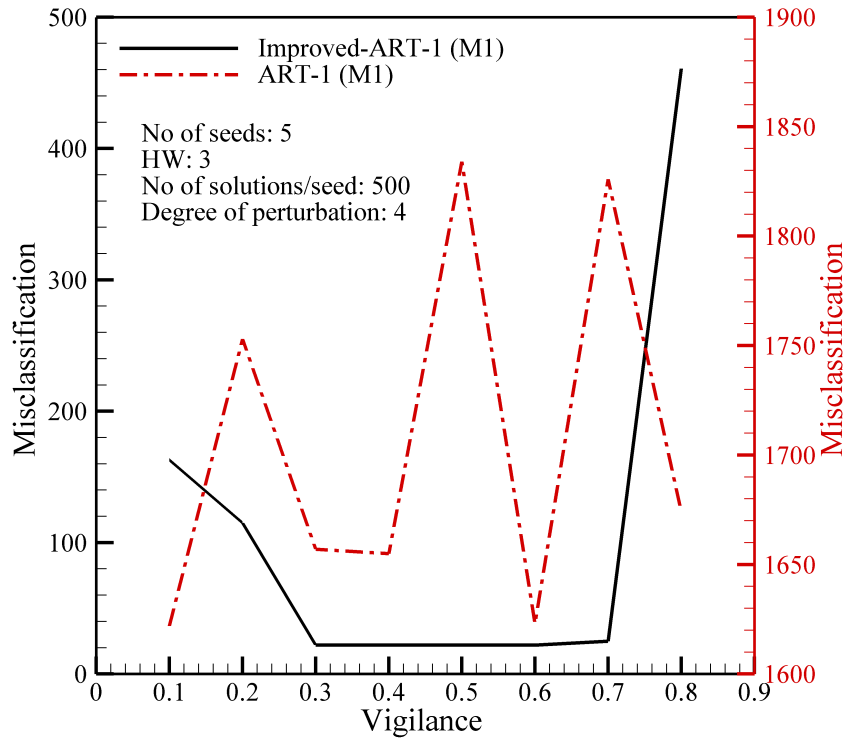


Figure 3.20: Change in number of misclassification with vigilance parameter for Improved-ART-1 for 40 objects.

3.6. Conclusion

The work presented in that chapter shows two binary conversion techniques using ordered permutations sequences. These new binary conversion techniques can be potential alternatives that can be successfully used for any ANN and optimization techniques where binary input is required. In addition, the proposed binary conversion methods have been numerically analyzed based on introduced performance evaluators that include misclassification, average distance, and homogeneity for both ART-1 and Improved-ART-1. The results show that binary conversion method-1 (M1) is comparatively better than methods-2 (M2) in both ART-1 and Improved-ART-1. Additionally, It is identified that the Improved-ART-1 using M1 outperforms ART-1.

Chapter 4

ANN Guided Genetic Algorithm

In the previous chapter, we have proposed that Improved-ART-1 outperforms while considering introduced binary conversion method-1. In this chapter, we have presented a technique for discriminating and clustering ordered permutations using ANN, which can be applied to develop a neural network-guided metaheuristic algorithm to solve complex problems. In the real world, many problems are having this kind of attribute. Typical examples include traveling salesman problems, flow shop scheduling, single row facility layout, and many other quadratic assignment problems. Therefore, we will consider a flexible flow shop scheduling problem among the other various problems by adopting the proposed Improved-ART-1 neural network with binary conversion method-1 guided GA. Besides, the detailed architecture of Improved-ART-1-guided GA has been presented.

In this chapter, at first, we have discussed a flexible flow shop scheduling problem; then, a detailed of GA has been described by presenting mathematical formulation, where solution representation, fitness evaluation, assignment rule, genetic operators, which include selection operator, crossover operator, and mutation operator are deliberated. Afterward, the architecture of Improved-ART-1 neural network-guided GA has been elaborated.

4.1. Flexible Flow Shop Scheduling

Flow shop scheduling problem (FSP) is an alternative technique in a manufacturing scheduling environment where a number of jobs are processed by an identical flow pattern on a specific number of machines. A flexible flow shop scheduling problem (FFSP) is a multi-stage manufacturing system where the stages are arranged in a line or u-shaped layout. Each stage can have more than one unrelated parallel machines. Parts get processed by moving from one stage to the next in a unidirectional manner. Moreover, FFSP is the generalization of the classical and simple flow shop scheduling problem with parallel machines at each stage (Morton and Pentico, 1993). FFSP is referred to as a group of machines connected in parallel and arranged into a number of stages coupled in series. There are a number of identical machines at each step, and one job is processed at a time in each machine at most. FFSP sometimes is also known as hybrid FSP, blended FSP, and multi-stage scheduling (Tang, L-X and Wu, 2002). Over the last three and a half decades, many researchers have studied the FFSP problem due to avoiding excessive use of physical resources and getting reasonable and efficient schedules in the manufacturing and production industries.

Different algorithms have been considered for FFSP since the 1990s. Brah and Hunsucker (1991) adopts Branch-and-Bound algorithm to reduce the makespan for FSP. Since FFSP is an NP-hard type problem, it is sometimes difficult to solve in polynomial times; therefore, heuristic optimization algorithms have been introduced to solve NP-hard type problems to achieve a near-optimum solution. Pan *et al.* (2008) developed a multi-object heuristic algorithm for flexible line scheduling without considering process waiting time. Khalouli *et al.* (2010) adopted an ant colony optimization technique to optimize the delay in hybrid FFSP. Akrami *et al.* (2006) considered GA and tabu search in FFSP to obtain a near-optimal solution. According to the statistics stated in the literature review chapter, GA

is dominating as an optimization algorithm, and the number of research papers have been exponentially published over the years. Besides, there is also a tendency to combine ANN and GA as a hybrid approach for FFSP for further optimization. Hence, a number of researches have been found where ANN was considered with GA to get the near-optimal solution for FFSP (Azadeh *et al.*, 2019; Rezaeian *et al.*, 2011; Akyol, 2004). Therefore, in this work, we have proposed ANN guided GA for FFSP.

4.2. Mathematical Formulation

Here, we have presented the details of mathematical formulation for the mixed-integer linear programming (MILP) model and problem description with necessary assumptions. The main attributes are acquired from (Defersha and Chen, 2012b; Ruiz *et al.*, 2008), where lot streaming was considered. However, in this particular case, lot streaming has been removed.

4.2.1. Assumption and Problem Description

The mathematical formulation has been considered for the FFSP problem, which consists of several stages to process multiple jobs. According to the FFSP, a known number of machines in each stage are connected in parallel, and each stage is connected in series. It means a job from a given stage can be allocated to one of the machines for processing. Besides, there is a sequence-dependent setup time for each job to process on each entitled machine. This sequence-dependent setup time may be estimated or unestimated on different stages. A machine processes each job at most at a time. The processing sequence and the assignment of these jobs on each allocated machine at each stage need to be determined where the objective function is to minimize the completion time of processing the last job in the sequence to optimize the makespan.

4.2.2. Notations

We have defined various notations to present the mathematical model considered from (Defersha and Chen, 2012b). These notations are explained as follows:

Indexes and Input Data

I	Total number of stages where the stages are indexed by i or $l = 1, 2, \dots, I$
M_i	Total number of machines in stage i where the machines are denoted by $m = 1, 2, \dots, M_i$
N	Total number of jobs where the jobs are represented by n or $p = 1, 2, \dots, N$
P_n	A set of pairs of stages (l, i) for job n , i.e., the processing of job n in stage l is followed by its processing in stage i
$T_{n,m,i}$	Total processing time for one unit of job n on machine m in stage i
Q_n	Batch size of job n
$R_{m,i}$	Number of maximum production runs of machine m in stage i where the production runs are indexed by r or $u = 1, 2, \dots, R_{m,i}$
$S_{m,i,n,p}$	Setup time on machine m in stage i for processing job n following the processing of job p on this machine; if $n = p$, the setup may be called minor setup
$A_{n,i}$	A binary data equal to 1 if setup of job n in stage i is attached (non-anticipatory), or 0 if this setup is detached setup (anticipatory)
$B_{n,i}$	A binary data equal to 1 if job n needs processing in stage i , otherwise 0

$D_{n,m,i}$	A binary data equal to 1 if job n can be processed on machine m in stage i , otherwise 0; $D_{n,m,i} \leq B_{n,i}$
$F_{m,i}$	The release date of machine m in stage i
Ω	Large positive number

Variables

Continuous Variables:

$c_{n,i}$	Completion time of the job n from stage i
$\hat{c}_{r,m,i}$	Completion time of the r^{th} run of machine m in stage i

Binary Variables:

$x_{r,m,i,n}$	Binary variable which takes the value 1 if the r^{th} run on machine m in stage i is for the job n , 0 otherwise,
$z_{r,m,i}$	A binary variable which equal to 1 if the r^{th} potential run of machine m in stage i has been assigned a job to process, 0 otherwise,
c_{max}	Makespan of the schedule.

4.2.3. MILP Model

The objective function and the constraints of the MILP mathematical model for the FFSP problem are given below:

Minimize:

$$Z = c_{max} \quad (4.1)$$

where. Z defines the objectives.

Subject to:

$$\hat{c}_{r,m,i} \geq c_{n,i} + \Omega \cdot x_{r,m,i,n} - \Omega ; \quad \forall(r, m, i, n) \quad (4.2)$$

$$\hat{c}_{r,m,i} \leq c_{n,i} - \Omega \cdot x_{r,m,i,n} + \Omega ; \quad \forall(r, m, i, n) \quad (4.3)$$

$$\hat{c}_{1,m,i} - Q_n \cdot T_{n,m,i} - S_{m,i,n,0} - \Omega \cdot x_{1,m,i,n} + \Omega \geq F_{m,i} ; \forall(m, i, n) \quad (4.4)$$

$$\begin{aligned} \hat{c}_{r,m,i} - Q_n \cdot T_{n,m,i} - S_{m,i,n,p} - \Omega(x_{r-1,m,i,p} + x_{r,m,i,n}) + 2\Omega &\geq \hat{c}_{r-1,m,i} ; \\ \forall(r, m, i, n, p) | r > 1 \end{aligned} \quad (4.5)$$

$$\begin{aligned} \hat{c}_{1,m,i} - Q_n \cdot T_{n,m,i} - S_{m,i,n,0} \cdot A_{n,i} - \Omega \cdot (x_{u,k,l,n} + x_{1,m,i,j,n}) + 2\Omega &\geq \hat{c}_{u,k,l} ; \\ \forall(u, k, m, l, i, n) | (l, i) \in P_n \end{aligned} \quad (4.6)$$

$$\begin{aligned} \hat{c}_{r,m,i} - Q_n \cdot T_{n,m,i} - S_{m,i,n,p} \cdot A_{n,i} - \Omega(x_{r-1,m,i,p} + x_{u,k,l,n} + x_{r,m,i,n}) + 3\Omega &\geq \hat{c}_{u,k,l} ; \\ \forall(u, r, k, m, l, i, n, p) | (l, i) \in P_n, \quad r > 1 \end{aligned} \quad (4.7)$$

$$\sum_{n=1}^N x_{r,m,i,n} = z_{r,m,i} ; \quad \forall(r, m, i) \quad (4.8)$$

$$z_{r+1,m,i} \leq z_{r,m,i} ; \quad \forall(r, m, i) | r < R_{m,i} \quad (4.9)$$

$$\sum_{m=1}^{M_i} \sum_{r=1}^{R_{m,i}} x_{r,m,i,n} = B_{n,i} ; \quad \forall(i, n) \quad (4.10)$$

$$x_{r,m,i,n} \leq D_{n,m,i} ; \quad \forall(r, m, i, n) \quad (4.11)$$

$$c_{max} \geq c_{n,i} ; \quad \forall(n, i) \quad (4.12)$$

$$x_{r,m,i,n}, z_{r,m,i}, \text{ and are binary} \quad (4.13)$$

The objective function in Eq. (4.1) is to minimize the schedule's makespan, which is equal to the completion time of the last job processed in the system. The constraints in Eqs. (4.2) and (4.3) together state that the completion time of the job n in stage i is equal to the completion time of the r^{th} run of machine m in stage i if this production run is assigned to that particular job. The starting time of the setup for the first run ($r = 1$) of machine m in stage i is given by $\hat{c}_{1,m,i} - Q_n \times T_{n,m,i} - S_{m,i,n,0}$ if the job n is assigned to this first run. This starting time cannot be less than the machine's release date, as enforced by the constraint in Eq. (4.4). The constraint in Eq. (4.5) is to enforce the requirement that the setup of any production run $r > 1$ of a given machine cannot be started before the completion time of run $r - 1$ of that machine. The constraint in Eq. (4.6) states that for any pair of stages $(l, i) \in P_n$, the setup or the actual processing of the first run on machine m in stage i may not be started before the completion time of run u of machine k in stage l , depending on whether the setup of product type n in stage i is non-anticipatory or anticipatory. This constraint is applied if run u of machine k in stage l and that of the first run of machine m in stage i are both assigned to the job n . The constraint in Eq. (4.7) is similar to that

in Eq. (4.6) except Eq. (4.7) is for run $r > 1$ of machine m in stage i . In this case, the sequence-dependent setup time has to be considered by taking into account the type of job that was processed in run $r - 1$ of machine m in stage i . The constraint in Eq. (4.8) controls the logical relations among the binary variables $x_{r,m,i,j,n}$, and $z_{r,m,i}$. The constraint in Eq. (4.9) is to enforce the logic that production runs $r + 1$ of a given machine can be assigned to a job if and only if run r of that machine is already assigned to another job. If job n is positive and it requires processing in stage i , then it should be assigned to one of the eligible machine in stage i . This is enforced by the constraints in Eqs. (4.10) and (4.11). Eq. (4.12) states that the makespan of the schedule, c_{max} , is greater or equal to the completion time of any job on any stage. At optimality, c_{max} takes the value of the completion time of the last job to be processed in the system. Eq. (4.13) is the integrality requirement.

4.3. Pure Genetic Algorithm

Genetic algorithm (GA) is one of the powerful search-based optimization tools which mimics the principle of genetics and natural selecting to find a near-optimal solution. GA is a branch of evolutionary computational algorithms. If the GA is not integrated with other algorithms or techniques, then it is said pure GA.

A basic flow chart of GA is presented in Fig.4.1. GA usually starts its process by randomly generating sets of solution strings known as the initial population. Each solution string of the population is defined as a chromosome, and each content of the chromosome is known as a gene (see Fig. 4.2). The pattern of gene organization in the chromosome is defined as solution encoding. After that, the fitness evaluation is calculated once the random generation of the initial population is finished. In the end, individual fitness is sorted and selected for

mating. Crossover and mutation are applied, and this process continues until a stopping condition is obtained or convergence is achieved.

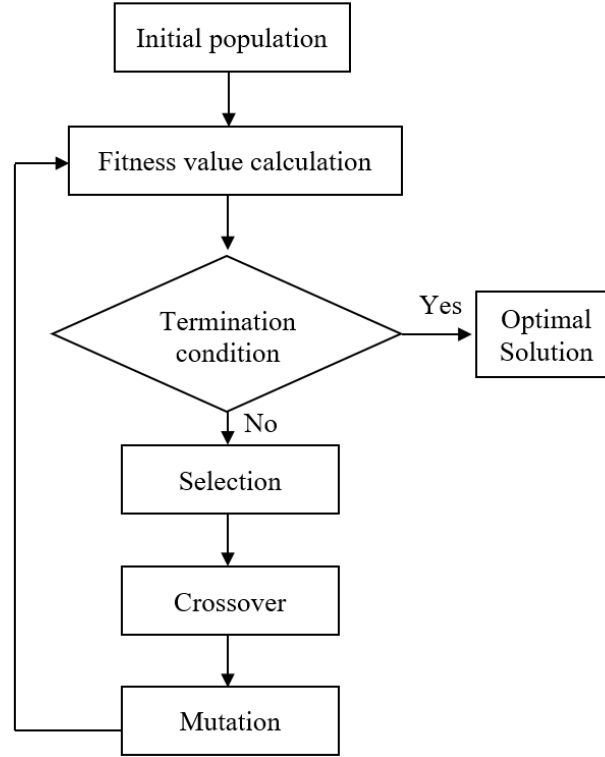


Figure 4.1: A basic flow chart of GA.

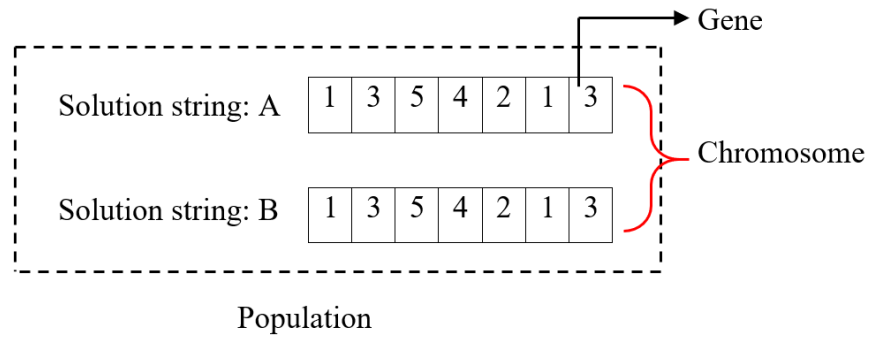


Figure 4.2: Schematic presentation of GA structure.

In this section, we have adapted the GA algorithm for FFSP problems from Defersha and Chen (2012b); Ruiz *et al.* (2008) to determine the optimal

or near-optimal solutions. Moreover, we have discussed solution representation of encoding, fitness evaluation, and operators that includes selection operator, crossover operator, and mutation operator.

4.3.1. Solution Representation

Solution representation is significant since the success and failure of implementing GA is mostly dependent on it (Yilmaz Eroglu and K"oksal, 2014). Therefore, solution representation is a critical stage in GA. Sometimes the solution representation is referred to as solution encoding, where each solution string will be encoded. Holland (1975) stated that in order to implement the GA successfully, it is necessary to design the solution encoding correctly. Binary-based and permutation-based are the two widely used solution encoding methods. The binary-based encoding includes only 0s and 1s, whereas permutation-based coding includes a series of numbers. The permutation-based encoding is very popular, particularly in machine scheduling problems.

For FFSP, a simple permutation-based encoding has been considered for the chromosomes. Such simple encoding has been adopted to guide the assignment and processing sequencing of the jobs on the parallel machines at every stage using some form of heuristics or priority rules. Figure 4.3 presents the solution representation of the proposed GA where we assume ten jobs in a randomly generated permutation (i.e., 3, 4, 1, 2, 9, 10, 6, 8, 7, 5). Each chromosome shows a permutation π of N jobs where N is the total number of jobs. In this chromosome, $\pi(s)$ is the gene at the s^{th} location, and it is a numeric value equal to the indices of the job at that location. For example, in Fig.4.3, when $\pi(s)$ equals to 1 then it represents job 3 whereas $\pi(2)$ represents job 4 and so on.

<i>location</i> →	s = 1	2	3	4	5	6	7	8	9	10
→ <i>n</i>	3	4	1	2	9	10	6	8	7	5

Figure 4.3: Solution representation.

4.3.2. Fitness Evaluation

Once the initial population is generated and all the chromosomes are encoded, it is required to decode to find out the makespan. This makespan is considered the fitness value of the chromosome. The aim is to obtain better fitness value as it gives lower makespan. Therefore, in order to calculate the best makespan for assignment and process sequencing of our proposed FFSP, we have considered an assignment rule adopted by (Defersha and Chen, 2012b; Ruiz and Maroto, 2006) by adjusting few things according to our problem statement. The assignment rules are adjusted from the literature as: (i) considering attached and detached setup time; (ii) release date of machines; (iii) skip some stages for certain jobs. Now, if we consider job n that is allocated for a machine m in stage i then the completion time $c_{n,i}$ is going to be calculated by:

Case 1: If job n is considered as the first job to be assigned on machine m and stage i is the first stage for this job n , then $c_{n,i}$ becomes $F_{m,i} + S_{m,i,n,0} + Q_n \cdot T_{n,m,i}$.

Case 2: If job p is the last job that is assigned on machine m and i is the first stage to be visited for job n , then $c_{n,i}$ becomes $c_{p,i} + S_{m,i,n,p} + Q_n \cdot T_{n,m,i}$.

Case 3: If job n is the first job to be assigned on machine m and this job n has to visit stage l right away before visits stage i , then $c_{n,i}$ becomes $\max\{F_{m,i} + (1 - A_{n,i}) \cdot S_{m,i,n,0}; c_{n,l}\} + Q_n \cdot T_{n,m,i} + A_{n,i} \cdot S_{m,i,n,0}$.

Case 4: If job p is the last job to be assigned on machine m , and job n has to visit stage l instantly before visits stage i , then $c_{n,i}$ becomes $\max\{c_{p,i} + (1 - A_{n,i}) \cdot S_{m,i,n,p}; c_{n,l}\} + Q_n \cdot T_{n,m,i} + A_{n,i} \cdot S_{m,i,n,p}$.

Assignment Rule

In assignment rule, we are considering sequence-dependent setup time, processing times of different units on parallelly connected machines, machine release date, and machine eligibility in order to calculate the completion time of each job in π that visits all the stages. We have adopted this assignment rule from the detailed work of (Ruiz and Maroto, 2006). The basic steps of this assignment rule are given as follows:

- Step 1.** Decode the sizes of each job for the considered chromosome and get the permutation π from the chromosome (as shown in Fig. 4.3). Initialize $s = 1$ and $i = 1$.
- Step 2.** Set $n = \pi(s)$.
- Step 3.** If $Q_n > 0$, go to Step 4; else go to Step 7.
- Step 4.** If $B_{n,i} = 1$, go to Step 5; else go to Step 6.
- Step 5.** Assign job n to one of the available machine m in stage i that can result in the earliest completion time $c_{n,i}$ of this job in this stage.
- Step 6.** If $i < I$, set $i = i + 1$ and go to Step 4; else go to Step 7.
- Step 7.** If $s < N$, set $s = s + 1$ and go to Step 2; else go to Step 8.
- Step 8.** Calculate $c_{max} = \max_{\forall(n,i)} c_{n,i}$. Here, c_{max} defines the smallest makespan/completion time of all jobs processed in the system of considered chromosome.

4.3.3. Genetic Operators

Once we have each chromosome's fitness value, then the next step is to choose the best fitness value. GA operators develop a more promising solution from the initial population to replace the less promising solution. Therefore, this is an

important stage in GA. There are basically three operators, such as selection, crossover, and mutation operators, that take part in this process.

4.3.3.1 Selection Operator

The selection operator is the first operator used for reproduction, which means this operator selects the chromosome, that will take part in the next iteration in GA. There are many selection operators used in GA. [Sivanandam and Deepa \(2007\)](#) stated that the selection operators' main objective is to determine the best fitness solution string that will reproduce a new solution string with enhanced fitness. The selection operator takes part first in operation before crossover and mutation. In this thesis work, we have considered k -way tournament and roulette wheel-based on linear ranking selection operators. The individual k is chosen randomly in k -way tournament and can be adjusted. The k individual with the highest fitness value becomes the winner because it has the smallest complete time/makespan. This process is continued until the number of chosen individuals is equal to the size of the population.

In a roulette wheel based on linear ranking selection, the following steps are considered to adopt this operator in GA:

Step 1. Determine the fitness value of each solution string in the population.

Step 2. Sort out the population's solutions strings according to its fitness value and then rank them.

Step 3. The selection probability is fixed according to its rank for each of the chromosome.

Step 4. After that, all the chromosomes are accommodated on the roulette wheel according to their ranks.

Step 5. Each chromosome is assigned a specific segment in the wheel. This segment is proportional to its respective rank value. The higher the value of the

rank, the bigger the segment size in the wheel is.

Step 6. After placing all the fitness values, the wheel is virtually rotated. The selection process is completed when this rotation stops at a particular segment, which represents a specific fitness value of a solution string in the population. This selection process is continued until the required segment is selected. The segment with high rank has more probability to be selected in this process.

4.3.3.2 Crossover Operator

Once the selection operator finishes its selection, the new population's chromosome is randomly paired by crossover. The crossover operator is widely used in GA. A crossover operator is applied to each new pair of the chromosome to mix its features and form a new child with a high fitness value (Sivanandam and Deepa, 2007; Defersha and Chen, 2012b; Talbi, 2009). There are different types of crossover operators considered in the literature (Defersha and Chen, 2012a).

In this work, we have adopted a single-point crossover (SPC). Figure 4.4 presents SPC-1, where the exchange took place at the left-hand side. However, this exchange will be done on the right-hand side in SPC-2. The crossover operation has been accomplished in the following manner:

Step 1. Firstly, the crossover point is arbitrarily chosen from each parent. Then, all genes from the left-hand side of the crossover point of both parents are copied to generate two respective child and genes from the right side of the crossover point will be removed (see Fig. 4.4 (a)).

Step 2. The childs are interchanged between the parents i.e., child 1 goes to parent 2 and child 2 comes to parent 1. At last, the missing jobs are placed in their relative order of the other parent (see Fig. 4.4 (b)).

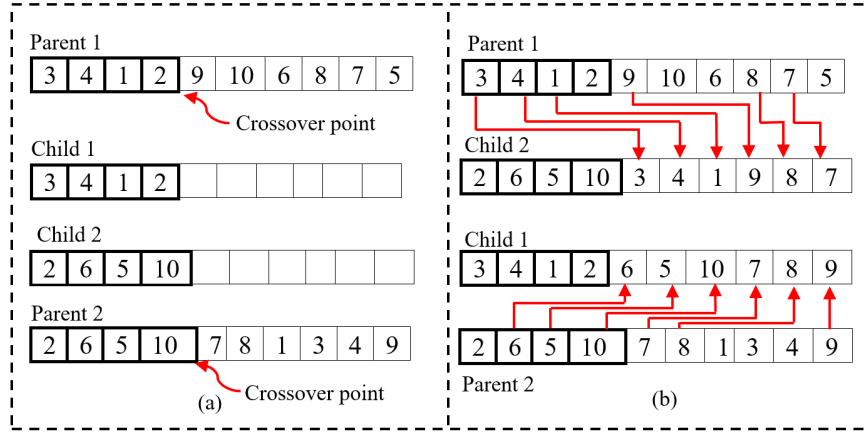


Figure 4.4: Presentation of single point crossover-1.

4.3.3.3 Mutation Operator

The mutation operator is another genetic operator and is deployed to the newly formed child after crossover operation. It is also known as a perturbation operator. The mutation operator changes a little bit of the new child chromosomes to achieve a better result. Many mutation operators are used in GA, such as swap mutation, shift mutation, flip mutation, bit string mutation, scramble mutation, inversion mutation, Random Operation Assignment Mutation (ROAM), Intelligent Operations Assignment Mutation (IOAM), Operations Sequence Shift Mutation (OSSM) (Defersha and Chen, 2012a).

In our work, we have used multi-shift mutation and multi-swap mutation. In swap mutation, we have randomly chosen two positions from the chromosome and then interchange its value. In Fig. 4.5, we have randomly selected 2 and 3, and after swapping, these 2 and 3 are interchanged their place. Swap mutation is widely used in permutation-based encoding.



Figure 4.5: An example of swap mutation.

Figure 4.6 illustrates a simple example of shift mutation. In shift mutation, we have randomly chosen two positions from the chromosome. Then one of the positions is placed after another chosen position, and the rest positions will be shifted accordingly.

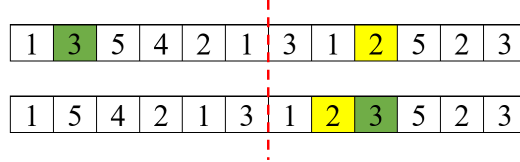


Figure 4.6: An example of shift mutation.

4.4. Proposed ANN-guided GA

Here, we are going to propose an architecture for developing ANN-guided GA. The Improved-ART-1 using binary conversion M1 has been considered as ANN here since Improved-ART-1 with M1 shows better performance. Figure 4.7 shows the architecture of the proposed ANN-guided GA to solve any problem whose solution can be represented as an ordered permutation. For example, flow shop scheduling, traveling salesman problem, single row facility layout, etc. can be the proposed ANN-guided GA technique's protentional application fields. The architecture has three major components: (1) Genetic Algorithm, (2) Improved-ART-1 Neural Network, and (3) Solution Warehouse/Cluster Management. GA evolves a population of solutions, in which each individual is an ordered permutation. It then periodically (every certain number of generations) sends a copy of the entire population of solutions to the ART module. As each solution passes through the ART module, it is first converted to a binary vector using our proposed binary conversion method-1 and then presented to the Improved-ART-1 neural network. The ART neural network assigns a cluster-ID to each individual.

Once each individual is assigned a cluster-ID, the population is sent to the Solution Warehouse/Cluster Management (SWCM) module. SWCM places each individual from the incoming population in its respective cluster and dynamically inserts it based on its fitness value. The more fit individuals are placed at the top of the cluster. Here, it is important to mention that each cluster in the SWCM is a double linked list having an iterator and an insertion mechanism. Thus, automatic sorting is accomplished with minimal computational effort. Moreover, SWCM will not store an individual solution if it is not unique. To this end, clusters in SWCM do not contain duplicates. In each cycle of communication of the three modules (GA, ANN, and SWCM), the SWCM flags a cluster if it absorbs a large percentage of the incoming population. If a cluster is flagged for more than a specified number of times, SWCM will send a signal to the GA to penalize individuals that belong to that flagged cluster. The penalty is being accomplished by replacing the individual solution that belongs to a repeatedly flagged cluster by a randomly generated individual. If new clusters are not being created and all the clusters are flagged, SWCM will reset the flag counter of each cluster to zero, and the search will continue in a similar manner until another reset is needed.

The intervention of GA by SWCM stated above will prevent the algorithm from stagnating in a given region of the search space. Hence, the intervention promotes exploration of the solution space. However, whenever exploitation around the best solutions is needed, the SWCM sends the best individual from clusters having small average makespan to the GA and will stop flagging clusters until it is time to restart exploration.

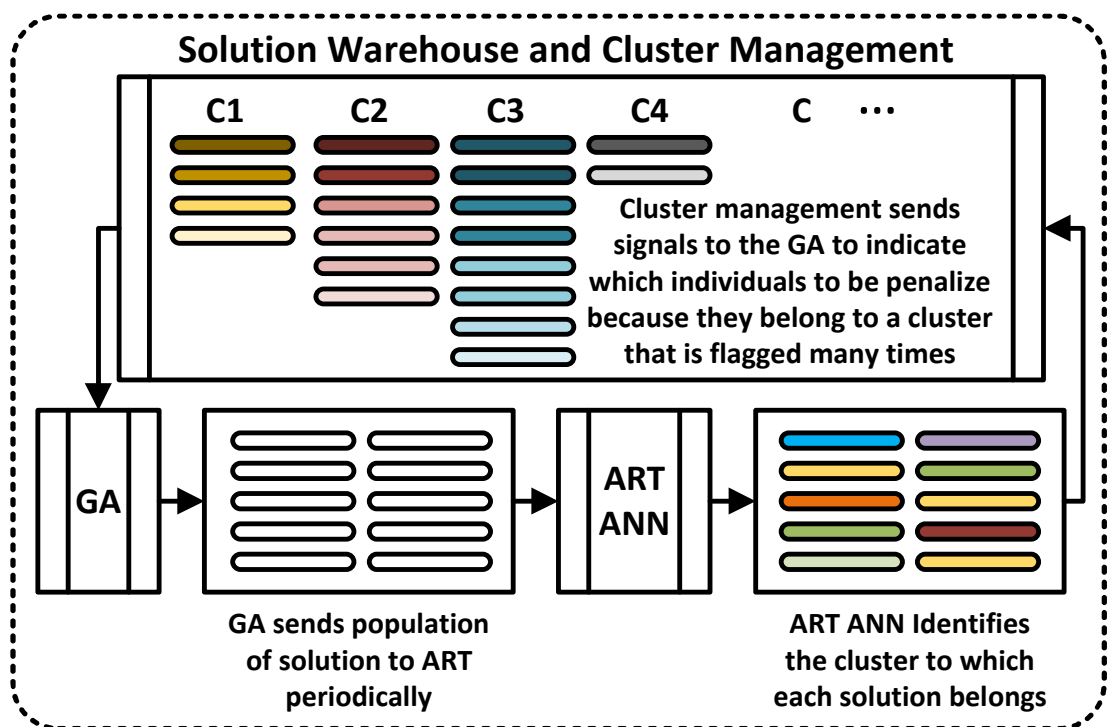


Figure 4.7: Architecture of the Improved-ART-1 ANN-guided GA.

Chapter 5

Numerical Investigation

Noticeable efforts have been made to develop Artificial Neural Network (ANN) guided metaheuristics, predominantly Genetic algorithm (GA), over the last few decades, which can be a potential alternative to traditional optimization techniques for complex problems. Since literature is limited on ANN-guided GA in manufacturing; besides, no literature is found on potential applications in ART neural network-guided GA. Therefore in this chapter, we would like to present the computational performance of the proposed algorithm, i.e., ART-guided GA using binary conversion method 1 in order to address the FFSP. Furthermore, we have presented a comparison between pure GA and Branch-and-Cut (BC) algorithm using IBM ILOG CPLEX solver for a small prototype problem. In the end, a comparison has been made between the pure GA and the proposed ART-guided GA algorithm for a series of large problems to prove the effectiveness of proposed algorithms. The following sections will give more details of the investigated results.

5.1. Prototype Problem

Initially, a small prototype example of FFSP is considered for the MILP model. This prototype problem consists of eight jobs (n) that need to be processed in a total of four stages (i) using a maximum of two machines (m) in each stage. Table 5.1, 5.2, and 5.3 show the details of the small problem considered for FFSP. For instance, according to Table 5.1, each stage is assigned two parallel machines, i.e., the work assignment can be done using one of the machines depending on the calculated optimum processing time. We have also taken into account of machine release date $F_{m,i}$ in the proposed prototype problem. Machine release date is important in the production environment (Ruiz *et al.*, 2008) since the selection of the available machine can be influenced by the release date.

Table 5.1: A small example of case study consists of two machines and four stages.

Stage (i)	Number of parallel machine (M_i)	Machine (m)	Release date of machine m at stage i ($F_{m,i}$)
1	2	1	0
		2	40
2	2	1	40
		2	80
3	2	1	0
		2	120
4	2	1	80
		2	240

Table 5.2 illustrates the processing time required by the machines, where a total of eight jobs have been considered with various batch sizes that need to be completed at different stages. For example, job 1 (n) is completed in three stages (i.e., stage 1, 2, 4, and skipped stage 3), which involves three operations (o) considering a batch size (Q_n) of 20. Here, $A_{n,i}$ indicates the state of the

machine as a setup property. If $A_{n,i}$ is 1 then this setup is called attached or non-anticipatory. It also indicates that the machine setup will be determined once the job comes up at the machine. On the other hand, when $A_{n,i}$ is 0, it is called detached or anticipatory, which indicates that the setup of the machine must be completed before arriving of the job at the machine. In the proposed case study, we have considered both attached and detached setup conditions. The data for the sequence-dependent setup time of each eligible machine for different jobs at various stages are presented in Table 5.3 and Table 5.4.

Table 5.2: The case study includes eight jobs with different batch sizes and considered processing time for each eligible machine.

Job	Batch size	Operation	Stage	Setup Property	Processing time
(n)	(Q_n)	(o)	(i)	$(A_{n,i})$	on eligible machine, (m) , $(m, T_{n,m,i})$
1	20	1	1	0	(1, 6)
		2	2	1	(1, 4)(2, 3.5)
		3	4	0	(1, 6)(2, 6.5)
2	50	1	1	0	(1, 4.5)(2, 4)
		2	2	1	(1, 6.5)(2, 5.5)
		3	3	1	(1, 6.5)(2, 6.5)
3	20	1	1	1	(1, 3.5)(2, 4)
		2	2	0	(1, 3)(2, 3)
		3	3	1	(1, 3)(2, 4.5)
		4	4	1	(2, 5.5)
4	20	1	1	1	(1, 4)(2, 3)
		2	2	0	(1, 4.5)(2, 4)
		3	4	0	(2, 3.5)
5	40	1	1	0	(1, 4)(2, 3)
		2	2	1	(1, 3)(2, 4.5)
		3	3	0	(1, 3.5)(2, 4)
		4	4	1	(1, 5)
6	50	1	1	1	(1, 4)
		2	2	1	(1, 5)(2, 5.5)
		3	3	0	(1, 5.5)(2, 6.5)
7	30	1	1	0	(1, 5)(2, 6)
		2	2	1	(1, 6.5)(2, 5.5)
		3	4	1	(1, 3)
8	30	1	1	1	(1, 6.5)(2, 6.5)
		2	2	0	(1, 3.5)(2, 3.5)
		3	3	1	(1, 5)(2, 5)
		4	4	0	(1, 5)(2, 6.5)

Table 5.3: Setup time for each eligible machine for each job (Continued on next page).

Job (n)	Operation (o)	Stage (i)	Eligible machine, (m) (i.e., $D_{n,m,i} = 1$)	$S_{m,i,n,0}$ Setup time (if the operation is the first to be processed)	$(p, S_{m,i,n})$ Setup time if job n follows job p
1	1	1	1	120	(2, 340)(3, 180)(4, 80)(5, 200)(6, 280)(7, 360)(8, 220)
	2	2	1	100	(2, 100)(3, 80)(4, 260)(5, 400)(6, 320)(7, 180)(8, 240)
	3	4	2	120	(2, 340)(3, 400)(4, 260)(5, 320)(6, 380)(7, 60)(8, 360)
			1	100	(5, 80)(7, 340)(8, 260)
2	1	1	2	40	(3, 300)(4, 240)(8, 260)
			1	80	(1, 320)(3, 240)(4, 300)(5, 140)(6, 340)(7, 260)(8, 380)
	2	2	2	60	(3, 380)(4, 320)(5, 120)(7, 80)(8, 260)
			1	60	(1, 240)(3, 140)(4, 100)(5, 300)(6, 280)(7, 60)(8, 140)
			2	120	(1, 80)(3, 340)(4, 280)(5, 140)(6, 120)(7, 400)(8, 120)
			1	100	(3, 120)(5, 40)(6, 60)(8, 400)
3	1	1	2	80	(3, 260)(5, 140)(6, 100)(8, 140)
			1	60	(1, 100)(2, 360)(4, 300)(5, 240)(6, 200)(7, 260)(8, 380)
	2	2	2	120	(2, 260)(4, 240)(5, 260)(7, 280)(8, 260)
			1	80	(1, 60)(2, 260)(4, 60)(5, 120)(6, 100)(7, 400)(8, 100)
			2	100	(1, 100)(2, 400)(4, 400)(5, 240)(6, 120)(7, 280)(8, 320)
			1	100	(2, 120)(5, 200)(6, 400)(8, 260)
4	4	4	2	40	(2, 220)(5, 360)(6, 200)(8, 240)
			2	40	(1, 380)(4, 140)(8, 160)
	1	1	1	40	(1, 260)(2, 180)(3, 380)(5, 220)(6, 200)(7, 320)(8, 260)
			2	60	(2, 400)(3, 360)(5, 80)(7, 280)(8, 300)
	2	2	1	120	(1, 100)(2, 60)(3, 140)(5, 300)(6, 400)(7, 280)(8, 360)
			2	60	(1, 120)(2, 40)(3, 240)(5, 280)(6, 40)(7, 280)(8, 60)
	3	4	2	40	(1, 260)(3, 320)(8, 180)
			2	40	

Table 5.4: continued from previous page Table 5.3.

Job n	Operation o	Stage i	Eligible machine m (i.e., $D_{n,m,i} = 1$)	$S_{m,i,n,0}$ Setup time (if the operation is the first to be processed)	$(p, S_{m,i,n})$ Setup time if job n follows job p
5	1	1	1	100	(1, 140)(2, 260)(3, 120)(4, 160)(6, 120)(7, 120)(8, 380)
			2	120	(2, 160)(3, 240)(4, 180)(7, 60)(8, 100)
	2	2	1	100	(1, 140)(2, 140)(3, 160)(4, 340)(6, 160)(7, 320)(8, 120)
	3	3	2	120	(1, 240)(2, 400)(3, 220)(4, 100)(6, 320)(7, 80)(8, 380)
			1	80	(2, 60)(3, 320)(6, 120)(8, 280)
			2	60	(2, 220)(3, 280)(6, 160)(8, 220)
	4	4	1	80	(1, 300)(7, 220)(8, 180)
	1	1	1	40	(1, 140)(2, 120)(3, 200)(4, 280)(5, 200)(7, 180)(8, 320)
6	2	2	1	100	(1, 160)(2, 60)(3, 340)(4, 300)(5, 100)(7, 160)(8, 280)
			2	80	(1, 180)(2, 400)(3, 260)(4, 220)(5, 40)(7, 280)(8, 40)
	3	3	1	100	(2, 200)(3, 220)(5, 280)(8, 320)
			2	40	(2, 340)(3, 240)(5, 40)(8, 100)
7	1	1	1	80	(1, 140)(2, 400)(3, 240)(4, 300)(5, 220)(6, 80)(8, 300)
			2	80	(2, 140)(3, 280)(4, 200)(5, 120)(8, 240)
	2	2	1	80	(1, 100)(2, 260)(3, 160)(4, 340)(5, 260)(6, 60)(8, 280)
	3	4	2	80	(1, 400)(2, 320)(3, 260)(4, 60)(5, 340)(6, 320)(8, 80)
			1	40	(1, 340)(5, 300)(8, 380)
	1	1	1	40	(1, 60)(2, 320)(3, 400)(4, 260)(5, 120)(6, 380)(7, 280)
			2	80	(2, 280)(3, 180)(4, 400)(5, 120)(7, 380)
	2	2	1	100	(1, 320)(2, 320)(3, 220)(4, 400)(5, 360)(6, 320)(7, 240)
8	3	3	2	80	(1, 240)(2, 160)(3, 200)(4, 260)(5, 360)(6, 60)(7, 140)
			1	120	(2, 220)(3, 320)(5, 240)(6, 100)
			2	80	(2, 320)(3, 280)(5, 60)(6, 200)
	4	4	1	40	(1, 300)(5, 60)(7, 120)
			2	120	(1, 40)(3, 260)(4, 300)

5.2. Typical Solution

A typical solution representation of the proposed prototype problem is illustrated in Fig. 5.1. The first row (numbered as 1, 2, ...,8) indicates the location of the gene of the chromosome. The solution encoding in Fig. 5.1 will be decoded using the decoding procedure described in chapter 4 (section 4.3.2). Besides, the completion time of FFSP will also be calculated based on the fitness evaluation rules explained in section 4.3.2 of chapter 4.

Location (<i>s</i>)							
1	2	3	4	5	6	7	8
Jobs (<i>n</i>)							
1	8	5	3	2	6	4	7

Figure 5.1: A typical solution representation for FFSP.

An example of the GA decoding procedure for the solution representation is presented in Table 5.5 and Table 5.6. Table 5.1, 5.2, and 5.3 are considered for the decoding procedure. The details of the GA decoding procedure that includes four cases (Case 1 to Case 4) is described in chapter 4 in section 4.3.2. At the end, the total completion time of each job at four different stages completed by each machine can be calculated until all the jobs are assigned and processed. Table 5.7 presents the completion time of each job completed at four stages using machine m_1 and machine m_2 , and the color indicates which machine completes the jobs faster.

Besides, Figure 5.2 shows a pictorial presentation of the Gantt chart of the schedule corresponding to the solution presented in Fig 5.1. Gantt chart indicates that the feature of the stated prototype problem, which needs a maximum makespan to complete the whole operation, is 1835 minutes. In the Gantt chart, there are some jobs where $A_{n,i}$ is 0, which indicates that the setup time of that particular job in that stage can be started earlier if possible before completion of

the corresponding job from the previous stage. According to this, the setup time of job n_7 on machine m_1 and job n_8 on machine m_2 at stage i_2 , job n_5 and job n_6 at stage i_3 on machine m_1 and machine m_2 respectively, and job n_1 on machine m_1 and job n_4 on machine m_2 at stage i_4 have been started earlier.

Table 5.5: Partial illustration of GA decoding steps for the solution representation in Fig. 5.1.

s	n	i	m	
1	1	1	1	Step 1: This is the 1 st stage for n_1 and the 1 st job to be processed on this machine, according to the conditions, Case 1 will be considered; Step 2: $C_{1,1}=F_{1,1}+S_{1,1,1,0}+Q_1 \times T_{1,1,1}=0+120+(20 \times 6)=240$ min
			2	Machine is not eligible for n_1 Decision: m_1 is considered at this stage for n_1 where completion time is 240 min.
		2	1	Step 1: This is the 1 st job assigned on this machine at stage i_2 , according to the condition Case 3 is applied; Step 2: $C_{1,2}=\max\{F_{1,2}+(1-A_{1,2}) \times S_{1,2,1,0}; C_{1,1}\}+Q_1 \times T_{1,1,2}+A_{1,2} \times S_{1,2,1,0}=\max\{40+(1-1) \times 100; C_{1,1}\}+20 \times 4+1 \times 100=\max\{40;240\}+180=420$ min
			2	Step 1: This is the 1 st job assigned on this machine at stage i_2 , according to the condition Case 3 is applied; Step 2: $C_{1,2}=\max\{F_{2,1}+(1-A_{1,2}) \times S_{2,2,1,0}; C_{1,1}\}+Q_1 \times T_{1,2,2}+A_{1,2} \times S_{2,2,1,0}=\max\{80+(1-1) \times 120; C_{1,1}\}+20 \times 3.5+1 \times 120=\max\{80;240\}+190=430$ min Decision: Since m_1 takes less time than m_2 therefore the completion time will be 460 min.
		3	1	Machine is not required
			2	Machine is not required
		4	1	Step 1: This is the 1 st job assigned on this machine at stage i_4 , according to the condition, Case 3 is applied; Step 2: $C_{1,4}=\max\{F_{1,4}+(1-A_{1,4}) \times S_{1,4,1,0}; C_{1,2}\}+Q_1 \times T_{1,1,4}+A_{1,4} \times S_{1,4,1,0}=\max\{80+(1-0) \times 100; C_{1,2}\}+20 \times 6+0 \times 100=\max\{180;420\}+120=540$ min
			2	Step 1: This is the 1 st job assigned on this machine at stage i_4 , according to the condition, Case 3 is applied; Step 2: $C_{1,4}=\max\{F_{2,4}+(1-A_{1,4}) \times S_{2,4,1,0}; C_{1,2}\}+Q_1 \times T_{1,2,4}+A_{1,4} \times S_{2,4,1,0}=\max\{240+(1-0) \times 40; C_{1,2}\}+20 \times 6.5+0 \times 40=\max\{240;420\}+130=550$ min Decision: Since m_1 takes less time than m_2 therefore the completion time 540 minutes. will be chosen.

This decoding process will be continued in the next Table 5.6 for job 8.

Table 5.6: Continuation of Table 5.5 for GA decoding steps of job 8 ($n=8$).

s	n	i	m			
2	8	1	1	Step 1: This is the 1 st stage for n_8 but not the 1 st job for this machine, according to the conditions, Case 2 will be considered; Step 2: $C_{8,1}=C_{1,1}+S_{1,1,1,1}+Q_8 \times T_{8,1,1}=240 + 60 + (30 \times 6.5) = 495$ min		
			2	Step 1: This is the 1 st stage for n_8 and the 1 st job to be processed on this machine, according to the conditions, Case 1 will be considered; Step 2: $C_{8,1}=F_{2,1}+S_{2,1,8,0}+Q_8 \times T_{8,1,1}=315$ min Decision: m_2 is considered at this stage for n_8 where completion time is 315 min.		
		2	1	1	Step 1: This is neither the 1 st stage for n_8 nor the 1 st job to be processed on this machine at stage i_2 , according to the conditions, Case 4 is applied; Step 2: $C_{8,2}=\max\{C_{1,2}+(1-A_{8,2}) \times S_{1,2,8,1}; C_{8,1}\}+Q_8 \times T_{8,1,2}+A_{8,2} \times S_{1,2,8,1}=\max\{420+(1-0) \times 320; C_{8,1}\}+30 \times 3.5+0 \times 320=\max\{420; 315\}+425=845$ min	
				2	Step 1: This is the 1 st job assigned on this machine at stage i_2 , according to the condition, Case 3 is applied; Step 2: $C_{8,2}=\max\{F_{2,2}+(1-A_{8,2}) \times S_{2,2,8,0}; C_{1,2}\}+Q_8 \times T_{8,2,2}+A_{8,2} \times S_{2,2,8,0}=420$ min Decision: Since m_2 takes less time than m_1 therefore the completion time will be 420 min.	
			3	1	1	According to the conditions, Case 3 is applied, where, $C_{8,3}=\max\{F_{1,3}+(1-A_{8,3}) \times S_{1,3,8,0}; C_{8,2}\}+Q_8 \times T_{8,1,3}+A_{8,3} \times S_{1,3,8,0}=690$ min
					2	According to the conditions, Case 3 is applied, where, $C_{8,3}=\max\{F_{2,3}+(1-A_{8,3}) \times S_{2,3,8,0}; C_{8,2}\}+Q_8 \times T_{8,2,3}+A_{8,3} \times S_{2,3,8,0}=650$ min Decision: Since m_2 takes less time than m_1 therefore the completion time will be 650 min.
	4	1	1	According to the conditions, Case 4 is applied, where, $C_{8,4}=\max\{C_{1,4}+(1-A_{8,4}) \times S_{1,4,8,1}; C_{8,3}\}+Q_8 \times T_{8,1,4}+A_{8,4} \times S_{1,4,8,1} = 990$ min		
			2	According to the conditions, Case 3 is applied, where, $C_{8,4}=\max\{F_{2,4}+(1-A_{8,4}) \times S_{2,4,8,0}; C_{8,4}\}+Q_8 \times T_{8,2,4}+A_{8,4} \times S_{2,4,8,0}=845$ min Decision: Since m_2 takes less time than m_1 therefore the completion time 845 minutes will be chosen.		
	This decoding process will be continued until all the jobs are assigned at each stage.					

Table 5.7: Completion time of each job that is processed by each machine at four stages.

Jobs (n)	Machines (m)	Stages (i)			
		1	2	3	4
1	1	Case 1: 240	Case 3: 420	NR	Case 3: 540
	2	NE	Case 3: 430	NR	Case 3: 550
8	1	Case 2: 495	Case 4: 845	Case 3: 690	Case 4: 990
	2	Case 2: 315	Case 3: 420	Case 3: 650	Case 3: 845
5	1	Case 2: 540	Case 4: 795	Case 3: 935	Case 4: 1435
	2	Case 2: 535	Case 4: 1095	Case 4: 1030	NE
3	1	Case 2: 410	Case 4: 975	Case 4: 1195	NE
	2	Case 2: 875	Case 4: 800	Case 4: 1130	Case 4: 1400
2	1	Case 2: 875	Case 4: 1480	Case 4: 1835	NR
	2	Case 2: 855	Case 4: 1470	Case 4: 2005	NR
6	1	Case 2: 810	Case 4: 1160	Case 4: 2330	NR
	2	NE	Case 4: 2145	Case 4: 1695	NR
4	1	Case 2: 1090	Case 4: 1650	NR	NE
	2	Case 2: 1315	Case 4: 1590	NR	Case 4: 1790
7	1	Case 2: 1540	Case 4: 1415	NR	Case 4: 1825
	2	Case 2: 1175	Case 4: 1815	NR	NE
m_1 m_2		NE = Not Eligible; NR = Not Required; Times in min.			

5.3. Solution of Prototype Problem (GA vs CPLEX)

A comparative study has been presented between pure GA using C++ and Branch-and-Cut(BC) algorithm using the IBM ILOG CPLEX solver package for the small-sized prototype problem. The compilation has been done on a 64-bit operating system-based computer with a speed of 3.5 GHz and has a RAM of 16GB. Figure 5.3 shows the results from the compilation of CPLEX solver for the prototype problem. It is evident from the graph that CPLEX solver could not find an optimum solution even after 69 hours of compilation and provide the results with a makespan of 1875 minutes where the optimality gap between the best integer solution and lower bound is 0.389. Figure 5.4 shows the performance of the pure GA, where it takes nearly 30 milli-seconds for convergence to determine the near-optimal solution with a makespan of 1785 minutes. The convergence graph gradually becomes stable in steady condition, which is the

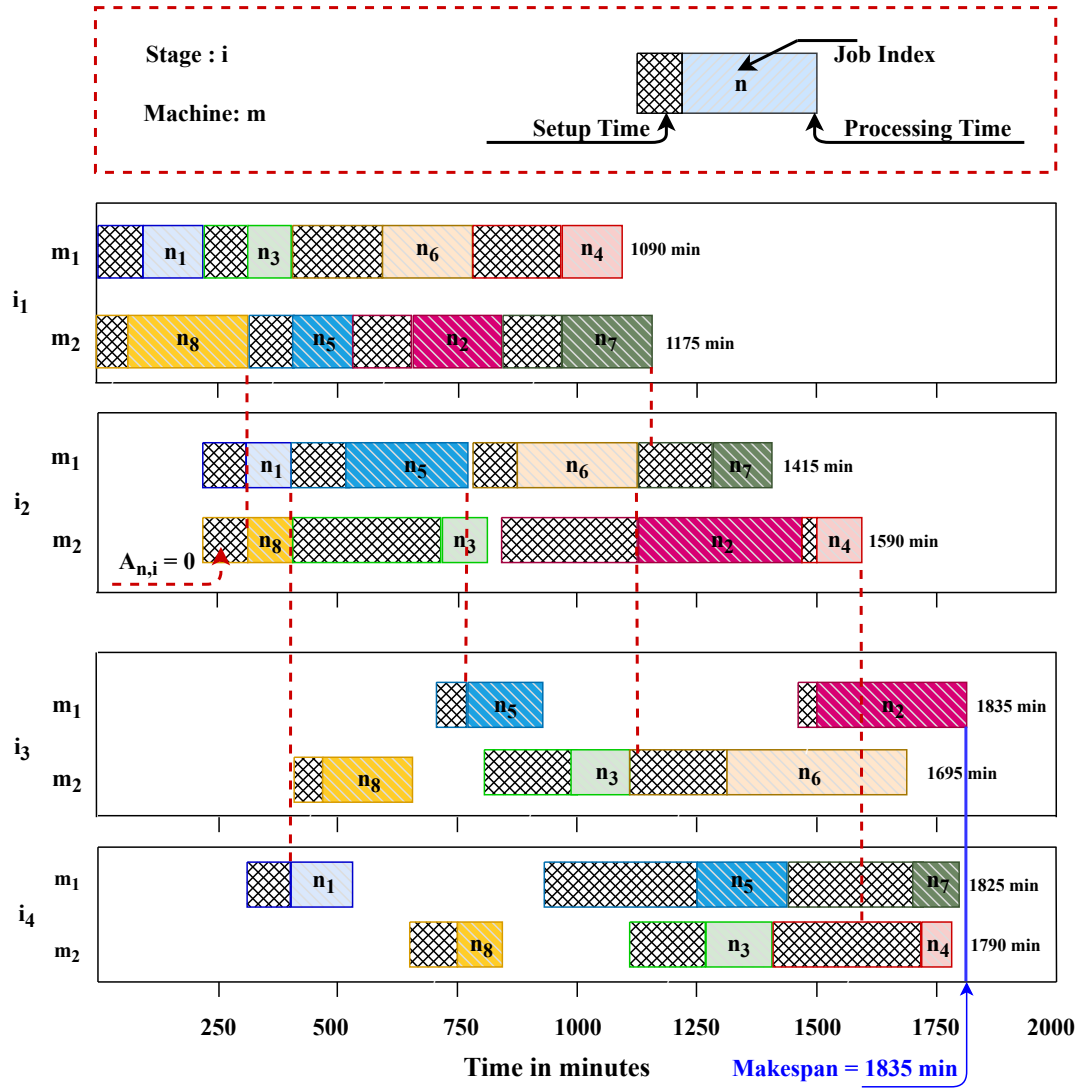


Figure 5.2: Gantt chart of machine scheduling for prototype problem solution.

optimal solution in that case. Therefore, it can be said that GA outperforms the BC algorithm in terms of computational time.

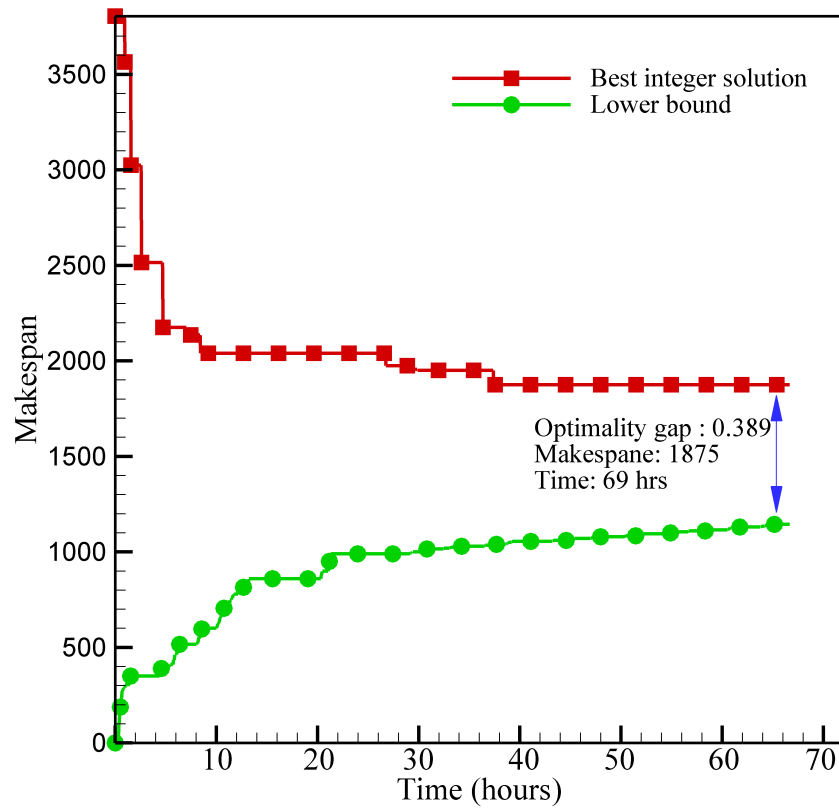


Figure 5.3: Optimality graph for prototype problem using BC CPLEX solver.

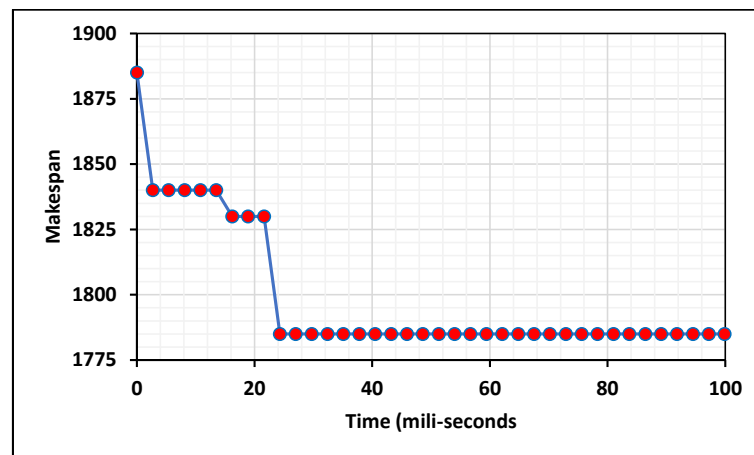


Figure 5.4: Convergence graph for prototype problem using pure GA.

5.4. GA and ANN-guided GA for Large Size Problems

It is evident from the results according to the section 5.3 that pure GA outperforms in terms of completion time compared to BC using IBM ILOG CPLEX solver for the small prototype problem. Therefore, in this section, we are going to compare between pure GA and our proposed algorithm ART neural network-guided GA for large size problems. Besides, six large problems have been considered for this comparative study, and these problems have been generated and solved using the 1st 20 test runs considering only Pure GA without ANN intervention and the last 20 with ANN-guided GA. Table 5.8 presents the general features of the large size problems with considered number of jobs, number of stages, maximum and minimum number of parallel machines, maximum and minimum number of batch sizes for each problem.

Table 5.8: Features of considered large size problems.

Problem No.	No of Jobs (n)	No of Stages (i)	No of Parallel Machine (m)	No of Batch Size (Q_n)
1	20	10	4 to 2	30 to 60
2	6	10	5 to 2	20 to 50
3	60	20	4 to 3	20 to 50
4	90	6	4 to 3	20 to 50
5	90	1	6 to 4	20 to 50
6	90	8	8 to 4	20 to 50

We present six case studies (i.e., large problems) where we apply the proposed ANN-guided GA and pure GA to solve a flexible flow shop scheduling problem (FFSP) with sequence-dependent setup time. As we know that FFSP is

a multi-stage manufacturing system where the stages are arranged in a line or u-shaped layout. Each stage can have more than one unrelated parallel machines. Parts get processed by moving from one stage to the next in a unidirectional manner. The problem is to determine an effective sequence of the jobs at the first stage so that all the jobs can be completed in a minimal amount of time (makespan). Thus, the solution to the problem is an ordered list of jobs. Each ordered list can be assigned a measure of goodness, which is makespan. Figure 5.5-(a) and -(b) show the convergence graph of the genetic algorithm without and with ANN-intervention, respectively. As it can be seen in this convergence graph, the GA with ANN found a solution with a lower makespan. The experiment was repeated twenty times by changing the seed of the random number generator. In many of the trials, the ANN-guided GA identifies solutions with lower makespan as it is depicted in Figure 5.6.

Similarly, the other five large problems have been investigated in order to find out the convergence and makespan for GA without and with ANN algorithms. It is clearly noticeable from the analysis that the convergence graph patterns for the GA algorithm are similar irrespective of the problem sizes. Furthermore, Figure 5.7, 5.8, 5.9, 5.10, and 5.11 prove that ANN-guided GA completes the process with lower makespan compared to pure GA in all considered problem sizes in each twenty trials. Therefore, the proposed ART neural network-guided GA performs better than pure GA.

The previous section and this result clarify that GA is capable of finding a solution in less time, but it cannot be guaranteed to have an optimal solution. The results observed from Figure 5.12 and 5.13 show that GA is exploring only certain clusters, which indicates the exploitation of GA because of previously explored regions by GA make searching options blind without exploring other

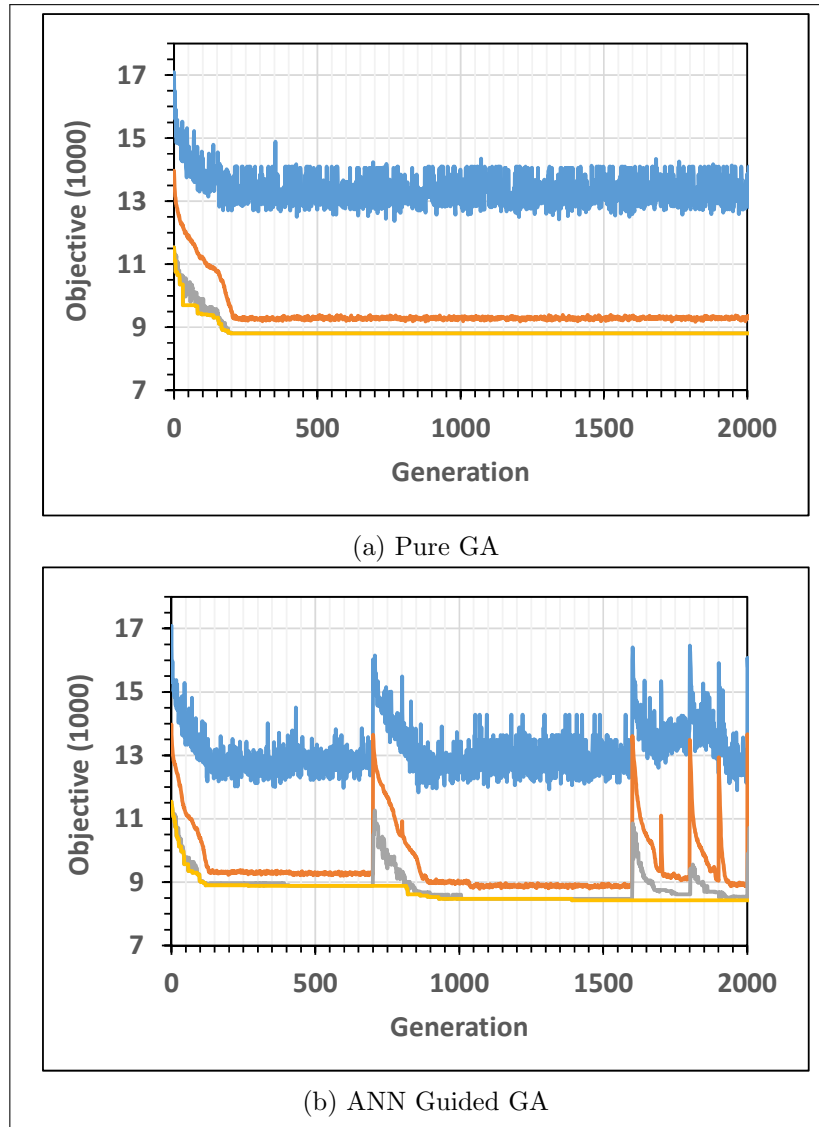


Figure 5.5: A typical convergence graphs (Blue - Worse Solution; Orange - Average; Gray - Good; Yellow - Best so far)

clusters where optimal solutions can be found. This is one of the limitations of GA. The proposed algorithm, ANN-guided GA architecture, combines GA, ART, and SWCM to overcome this problem of pure GA. According to section 4.4, the intervention of GA by SWCM will prevent the algorithm from stagnating in a given region of the search space. Hence, the intervention promotes exploration of the solution space to make other clusters to find the optimal solution. Therefore,

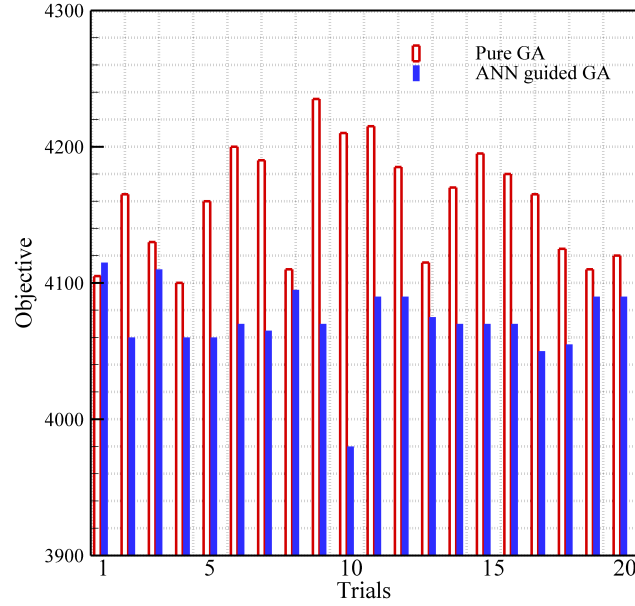


Figure 5.6: Comparison over multiple trails between GA and ANN-guided GA for problem 1.

the results show that ANN-guided GA has more exploration ability than pure GA and can find an optimal solution from searching almost all clusters.

Figure 5.12 and 5.13 illustrates comparison between the allocation of members in percentage for different large-size problems considered in Table 5.8 applying in pure GA and Improved ART-1 neural network-guided GA. These figures show how the cluster absorbs a large number of populations with and without ANN intervention. After 100,000 iterations, a total number of 330,000 populations are allocated in the different regions based on the applied algorithms. The bar graphs in Figure 5.12 show the number of members (%) that go to each cluster. From Figure 5.12 (a), (c), and (e), we can observe that one of the clusters absorbs a large percentage of the total population where many of them have almost nothing comparatively. Even some clusters have not been explored. In contrast, Figure 5.12 (b), (d), and (f) depict that most of the clusters have a

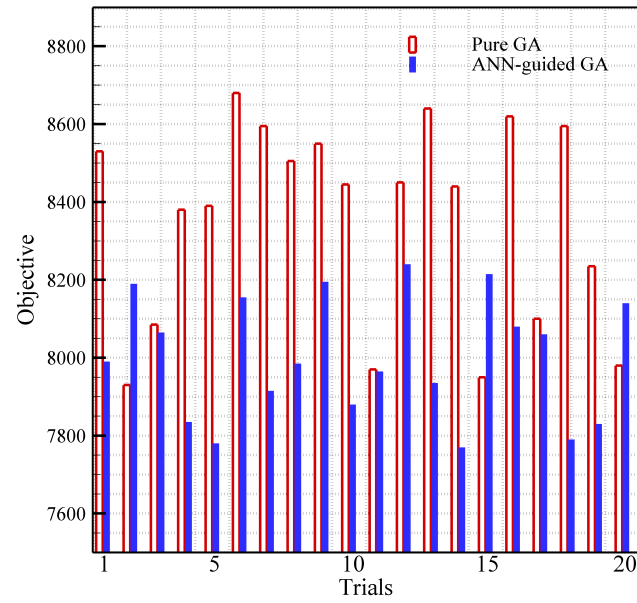


Figure 5.7: Comparison over multiple trails between GA and ANN-guided GA for problem 2.

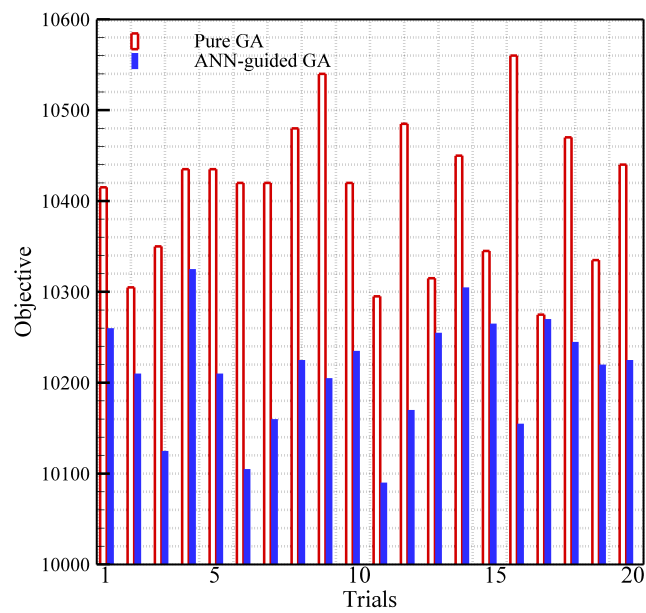


Figure 5.8: Comparison over multiple trails between GA and ANN-guided GA for problem 3.

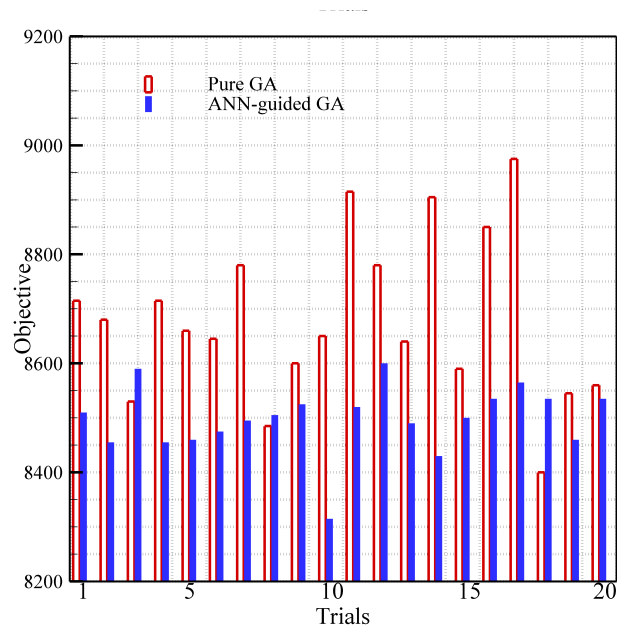


Figure 5.9: Comparison over multiple trails between GA and ANN-guided GA for problem 4.

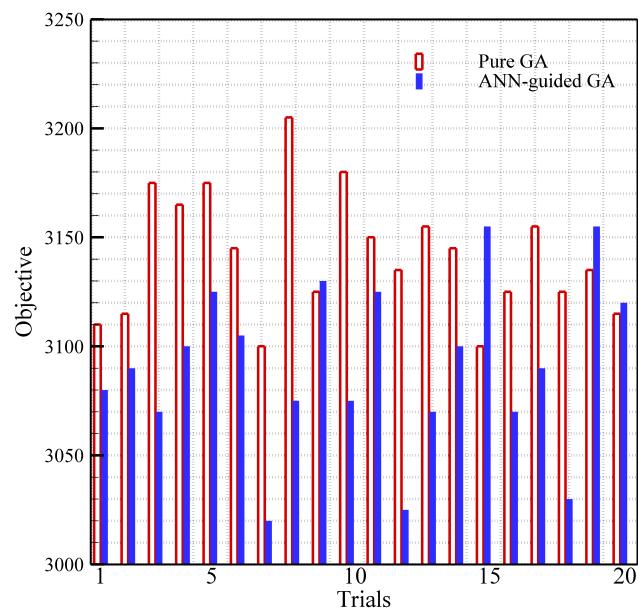


Figure 5.10: Comparison over multiple trails between GA and ANN-guided GA for problem 5.

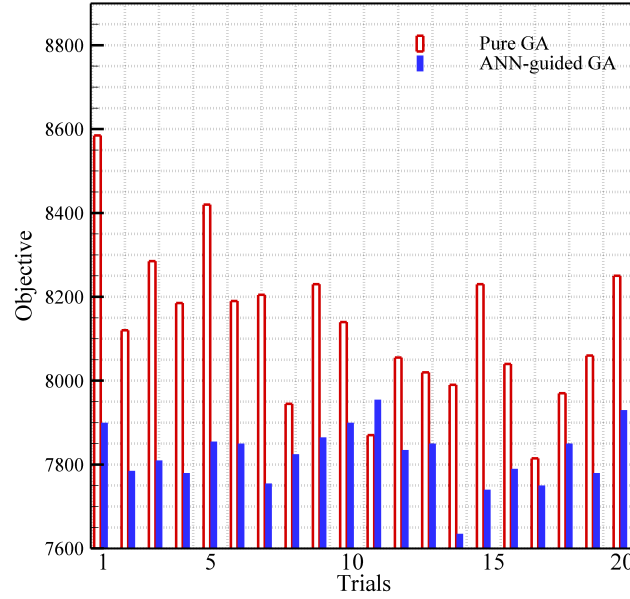


Figure 5.11: Comparison over multiple trails between GA and ANN-guided GA for problem 6.

reasonable number of members, which also indicates that many of the clusters have been searched thoroughly due to the intervention of ANN.

Figure 5.13 provides further insight into the analysis considering different problems where we can clearly see how pure GA has been stuck in one region of the search space. GA without ANN is being exploited in a certain cluster, i.e., GA is getting trapped in some regions where it is not clear that the result provided by GA is actually optimal one or we have still possibility to have a better result from other clusters which are unexplored yet (as shown in Fig. 5.13 (a), (c), and (e)). However, GA with ANN intervention, it is noticed that ANN helps to explore almost all the clusters of the search space (see in Fig. 5.13 (b), (d), and (f)).

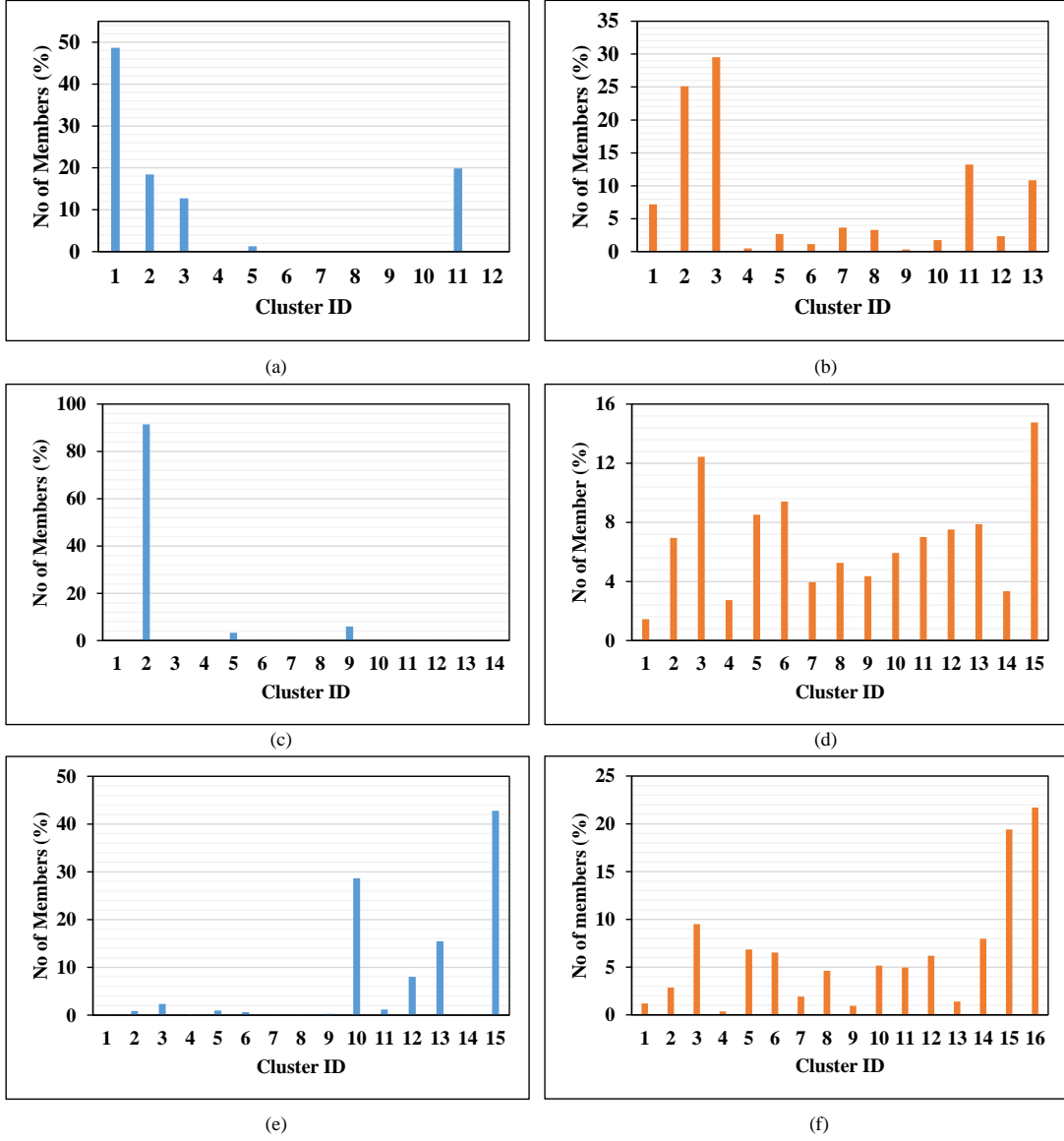


Figure 5.12: Formation of cluster by allocating the members based on the fitness value by GA (a), (c), (e) and ANN-guided GA (b), (d), (f) for problems 1, 3, and 5 respectively mentioned in Table 5.8.

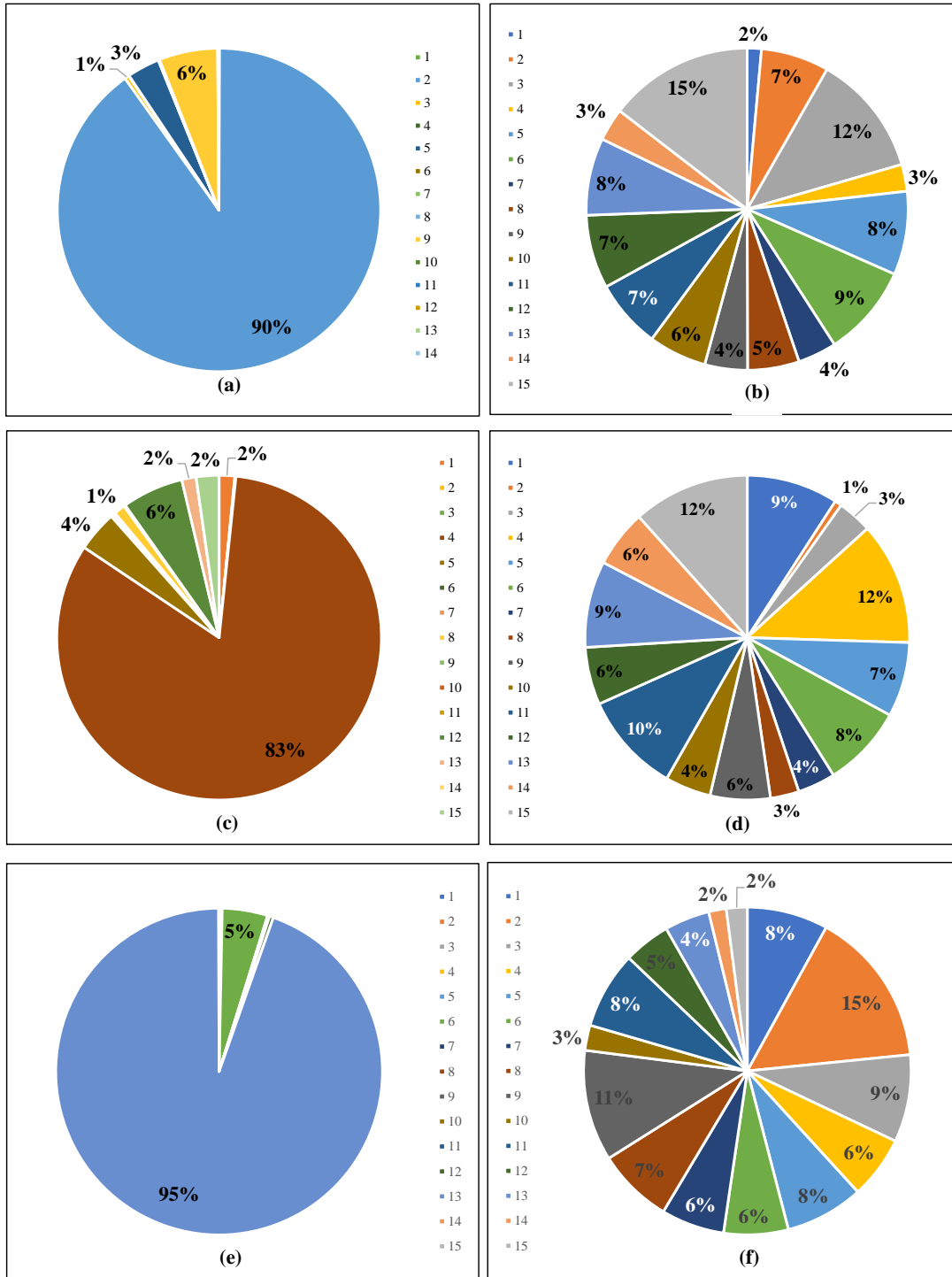


Figure 5.13: Allocation of population with and without intervention of ANN. (a), (c), (e) are the results of pure GA and (b), (d), (f) are for ANN-guided GA considering problems 4, 5, and 6 respectively mentioned in Table 5.8; the legends represent the cluster id.

Chapter 6

Research Outline

6.1. Summary and Conclusion

The research aims to improve makespan by adopting Improved ART-1-guided GA for flexible flow shop scheduling problems. As a summary of this thesis, we can conclude each chapter in the following ways, which will clearly describe our contribution and findings in this research.

Chapter 2: A comprehensive literature survey has been conducted in the field of GA and ANN-based manufacturing scheduling problems. This survey is focused on the hybridization of GA, especially with ANN. It is evident from the review that literature is limited in ANN-guided GA in metaheuristic applications, which motives us to do further research in this field.

Chapter 3: In this chapter, we have presented a technique for discriminating and clustering ordered permutation using ART-1 and Improved-ART-1. In the process, we introduced a new technique for converting ordered permutations to binary vectors to cluster them using ART-1. The proposed binary conversion methods are evaluated under varying three proposed performance indicators, i.e.,

misclassification, cluster homogeneity, and average distance. The performances of ART-1 and Improved-ART-1 have been investigated and compared based on the proposed binary conversion methods.

Chapter 4: Here, we have illustrated the details of the pure GA algorithm for the flexible flow shop scheduling problem (FFSP). In addition, a complete picture of the proposed ART-1 neural network guided GA architecture has been presented.

Chapter 5: In this chapter, we have analyzed the computational performance of the proposed algorithm, i.e., ART-guided GA adopting our developed binary conversion method to address the FFSP. Furthermore, we have presented a comparison between pure GA and a typical branch-and-cut algorithm for a small prototype problem where GA shows better performance. Finally, a comparison is made between pure GA and the proposed ART-guided GA algorithms using binary conversion method 1 (M1) for a series of large problems where ANN-guided GA outperforms compared to pure GA in every case. It proves the competence of our proposed algorithm for FFSP.

6.2. Future Research and Recommendations

The proposed ART-1 ANN-guided GA deals with manufacturing scheduling problems based on the binary matrix, which does not reflect the information regarding production volume, operation time, and operation sequence. Hence, alternative ANN can be considered to handle continuous real-time data (not only 0s and 1s) for compilation. We can focus on other ANN, such as Kohonen self-organizing map, and ART-2, can be focused on metaheuristic applications.

Here, our developed binary conversion based ART-1 neural network-guided GA is considered for FFSP. However, this proposed algorithm can also be applied to other problems, including job shop and other scheduling problems whose solution may not be pure permutation.

The future work also focuses on applying the architecture of ANN-Guided GA to non-permutation-based problems by developing equivalent binary representation techniques to their solutions. Besides, the application area of our developed ANN-guided GA should not be only limited to manufacturing scheduling problems but also can be considered for other fields such as, image recognition, pattern recognition, mobile robot control, signature verification, medical diagnosis, etc.

Bibliography

- Abdul-Razaq, T. S., Potts, C. N., and Van Wassenhove, L. N., 1990. A survey of algorithms for the single machine total weighted tardiness scheduling problem. *Discrete Applied Mathematics*, **26 (2-3)**, 235–253.
- Abe, M., Matsumoto, M., and Kuroda, C., 2000. An artificial neural network optimized by a genetic algorithm for real-time flow-shop scheduling. In: KES'2000. Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies. Proceedings (Cat. No.00TH8516). Vol. 1. pp. 329–332 vol.1.
- Agarwal, A., Colak, S., and Deane, J., 2010. NeuroGenetic approach for combinatorial optimization: An exploratory analysis. *Annals of Operations Research*, **174 (1)**, 185–199.
- Agarwal, A., Colak, S., and Erenguc, S., 2011. A Neurogenetic approach for the resource-constrained project scheduling problem. , **38 (1)**, 44–50.
- Agarwal, A., Colak, S., Jacob, V. S., and Pirkul, H., 2006. Heuristics and augmented neural networks for task scheduling with non-identical machines. *European Journal of Operational Research*, **175 (1)**, 296–317.
- Akrami, B., Karimi, B., and Moattar Hosseini, S. M., 2006. Two metaheuristic methods for the common cycle economic lot sizing and scheduling in flexible

- flow shops with limited intermediate buffers: The finite horizon case. *Applied Mathematics and Computation*, **183** (1), 634–645.
- Akyol, D. E., 2004. Application of neural networks to heuristic scheduling algorithms. In: *Computers and Industrial Engineering*. Vol. 46. Elsevier Ltd, pp. 679–696.
- Akyol, D. E. and Bayhan, G. M., 2007. A review on evolution of production scheduling with neural networks. *Computers and Industrial Engineering*, **53** (1), 95–122.
- Alippi, C., De Russis, C., and Piuri, V., 2003. A neural-network based control solution to air-fuel ratio control for automotive fuel-injection systems. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, **33** (2), 259–268.
- Anagnostopoulos, G. C. and Georgiopoulos, M., 2000. Hypersphere ART and ARTMAP for unsupervised and supervised, incremental learning. In: *Proceedings of the International Joint Conference on Neural Networks*. Vol. 6. IEEE, pp. 59–64.
- Ansari, A. and Bakar, A. A., 2015. A Comparative Study of Three Artificial Intelligence Techniques: Genetic Algorithm, Neural Network, and Fuzzy Logic, on Scheduling Problem. In: *Proceedings - 2014 4th International Conference on Artificial Intelligence with Applications in Engineering and Technology, ICAIET 2014*. Institute of Electrical and Electronics Engineers Inc., pp. 31–36.
- Asadzadeh, L., 2015. A local search genetic algorithm for the job shop scheduling problem with intelligent agents. *Computers and Industrial Engineering*, **85**, 376–383.

Bibliography

- Awodele, O. and Jegede, O., 2009. Neural Networks and Its Application in Engineering. In: Proceedings of the 2009 InSITE Conference. Informing Science Institute.
- Azadeh, A., Goodarzi, A. H., Kolaei, M. H., and Jebreili, S., 2019. An efficient simulation–neural network–genetic algorithm for flexible flow shops with sequence-dependent setup times, job deterioration and learning effects. *Neural Computing and Applications*, **31** (9), 5327–5341.
- Baker, K. R. and Trietsch, D., 2009. Principles of sequencing and scheduling. Wiley. com,
- Bhatt, N. and Chauhan, N. R., 2016. Genetic algorithm applications on Job Shop Scheduling Problem: A review. In: International Conference on Soft Computing Techniques and Implementations, ICSCIT 2015. Institute of Electrical and Electronics Engineers Inc., pp. 7–14.
- Brah, S. A. and Hunsucker, J. L., 1991. Branch and bound algorithm for the flow shop with multiple processors. *European Journal of Operational Research*, **51** (1), 88–99.
- Brito da Silva, L. E., Elnabarawy, I., and Wunsch, D. C., 2019. A survey of adaptive resonance theory neural network models for engineering applications. *Neural Networks*, **120**, 167–203.
- Burton, A. R. and Vladimirova, T., 1997. Utilisation of an adaptive resonance theory neural network as a genetic algorithm fitness evaluator. In: IEEE International Symposium on Information Theory - Proceedings. p. 209.
- Burton, A. R. and Vladimirova, T., 1998. Genetic Algorithm Utilising Neural Network Fitness Evaluation for Musical Composition — SpringerLink.

Bibliography

- Carpenter, G. A. and Grossberg, S., jan 1987. Neural dynamics of category learning and recognition: Attention, memory consolidation, and amnesia. *Advances in Psychology*, **42 (C)**, 239–286.
- Carpenter, G. A., Grossberg, S., and Reynolds, J. H., 1991. ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. *Neural Networks*, **4 (5)**, 565–588.
- Chien, C. F. and Chen, C. H., 2007. Using genetic algorithms (GA) and a coloured timed Petri net (CTPN) for modelling the optimization-based schedule generator of a generic production scheduling system. *International Journal of Production Research*, **45 (8)**, 1763–1789.
- Chou, F. D., 2009. An experienced learning genetic algorithm to solve the single machine total weighted tardiness scheduling problem. *Expert Systems with Applications*, **36 (2 PART 2)**, 3857–3865.
- Coello, C. A., 1999. An updated survey of evolutionary multiobjective optimization techniques: State of the art and future trends. In: Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999. Vol. 1. IEEE Computer Society, pp. 3–13.
- Colak, S. and Agarwal, A., 2005. Non-greedy heuristics and augmented neural networks for the open-shop scheduling problem. *Naval Research Logistics*, **52 (7)**, 631–644.
- Dagli, C. H. and Huggahalli, R., 1993. A Neural Network Approach to Group Technology — Neural Networks in Design and Manufacturing.
- Dagli, C. H. and Sittisathanchai, S., 1995. Genetic neuro-scheduler: A new approach for job shop scheduling. *International Journal of Production Economics*, **41 (1-3)**, 135–145.

- de Garis, H., 1994. An artificial brain ATR's CAM-Brain Project aims to build/evolve an artificial brain with a million neural net modules inside a trillion cell Cellular Automata Machine. *New Generation Computing*, **12** (2), 215–221.
- Deane, J., 2012. Hybrid genetic algorithm and augmented neural network application for solving the online advertisement scheduling problem with contextual targeting. *Expert Systems with Applications*, **39** (5), 5168–5177.
- Defersha, F. M. and Chen, M., 2012a. Jobshop lot streaming with routing flexibility, sequence-dependent setups, machine release dates and lag time. *International Journal of Production Research*, **50** (8), 2331–2352.
- Defersha, F. M. and Chen, M., 2012b. Mathematical model and parallel genetic algorithm for hybrid flexible flowshop lot streaming problem. *International Journal of Advanced Manufacturing Technology*, **62** (1-4), 249–265.
- Dorronsoro, B. and Pinel, F., jul 2017. Combining machine learning and genetic algorithms to solve the independent tasks scheduling problem. In: 2017 3rd IEEE International Conference on Cybernetics, CYBCONF 2017 - Proceedings. Institute of Electrical and Electronics Engineers Inc.
- Fazlollahtabar, H., Hassanzadeh, R., Mahdavi, I., and Mahdavi-Amiri, N., 2012. A genetic optimization algorithm and perceptron learning rules for a bi-criteria parallel machine scheduling. *Journal of the Chinese Institute of Industrial Engineers*, **29** (3), 206–218.
- Foo, Y. P. S. and Takefuji, T., 1988. Integer linear programming neural networks for job-shop scheduling. Publ by IEEE, pp. 341–348.
- Frankovič, B. and Budinská, I., 2000. Advantages and Disadvantages of Heuristic and Multi Agents Approaches to the Solution of Scheduling Problem. *IFAC Proceedings Volumes*, **33** (13), 367–372.

Bibliography

Fu, M. C., 1994. Optimization via simulation: A review.

Fuqing Zhao, Yi Hong, Dongmei Yu, and Yahong Yang, 2005. A Hybrid Approach Based on Artificial Neural Network and Genetic Algorithm for Job-shop Scheduling Problem. In: 2005 International Conference on Neural Networks and Brain. Vol. 3. IEEE, pp. 1687–1692.

Gonzalez, T. and Sahni, S., 1976. Open shop scheduling to minimize finish time. *Journal of the ACM (JACM)*, **23** (4), 665–679.

Grabowski, J. and Pempera, J., 2007. The permutation flow shop problem with blocking. a tabu search approach. *Omega*, **35** (3), 302–311.

Grossberg, S., 2013. Adaptive Resonance Theory: How a brain learns to consciously attend, learn, and recognize a changing world. *Neural Networks*, **37**, 1–47.

Gruau, F., Whitley, D., and Pyeatt, L., 1996. A Comparison between Cellular Encoding and Direct Encoding for Genetic Neural Networks. Tech. rep.

Haq, A. N., Ramanan, T. R., Shashikant, K. S., and Sridharan, R., 2010. A hybrid neural network-genetic algorithm approach for permutation flow shop scheduling. *International Journal of Production Research*, **48** (14), 4217–4231.

Hemanth, D. J., Selvathi, D., and Anitha, J., 2010. Application of Adaptive Resonance Theory Neural Network for MR Brain Tumor Image Classification. *International Journal of Healthcare Information Systems and Informatics*, **5** (1), 61–75.

Holland, J. H., 1975.

URL <https://mitpress.mit.edu/books/adaptation-natural-and-artificial-systems>

Hopfield, J. J. and Tank, D. W., 1985. "Neural" computation of decisions in optimization problems. *Biological Cybernetics*, **52** (3), 141–152.

- Huang, W. and Gao, L., 2020. A Time Wave Neural Network Framework for Solving Time-Dependent Project Scheduling Problems. *IEEE Transactions on Neural Networks and Learning Systems*, **31** (1), 274–283.
- Inthachot, M., Boonjing, V., and Intakosum, S., 2016. Artificial Neural Network and Genetic Algorithm Hybrid Intelligence for Predicting Thai Stock Price Index Trend. , .
- Jackson, J. R., 1955. Scheduling a production line to minimize maximum tardiness. Tech. rep., DTIC Document.
- Jain, A. K., Mao, J., and Mohiuddin, K. M., 1996. Artificial neural networks: A tutorial.
- Johnson, E. G., Kathman, A. D., Hochmuth, D. H., Cook, A. L., Brown, D. R., and Delaney, W. F., 1993. Advantages of genetic algorithm optimization methods in diffractive optic design. Vol. 10271. SPIE, p. 1027105.
- Johnson, S. M., 1954. Optimal two-and three-stage production schedules with setup times included. *Naval research logistics quarterly*, **1** (1), 61–68.
- Kaczmarczyk, W., 2011. Proportional lot-sizing and scheduling problem with identical parallel machines. *International Journal of Production Research*, **49** (9), 2605–2623.
- Kanada, Y., 2016. Optimizing neural-network learning rate by using a genetic algorithm with per-epoch mutations. In: Proceedings of the International Joint Conference on Neural Networks. Vol. 2016-October. Institute of Electrical and Electronics Engineers Inc., pp. 1472–1479.
- Kasap, N. and Agarwal, A., 2012. Augmented neural networks and problem structure-based heuristics for the bin-packing problem. *International Journal of Systems Science*, **43** (8), 1412–1430.

- Khalouli, S., Ghedjati, F., and Hamzaoui, A., 2010. A meta-heuristic approach to solve a JIT scheduling problem in hybrid flow shop. *Engineering Applications of Artificial Intelligence*, **23** (5), 765–771.
- Kumar, R., Aggarwal, R. K., and Sharma, J. D., 2013. Energy analysis of a building using artificial neural network: A review.
- Lee, H. C. and Dagli, C. H., 1997. A parallel genetic-neuro scheduler for job-shop scheduling problems. *International Journal of Production Economics*, **51** (1-2), 115–122.
- Li, Y. and Chen, Y., 2009. Neural network and genetic algorithm-based hybrid approach to dynamic job shop scheduling problem. In: Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics. pp. 4836–4841.
- Lo, Z. P. and Bavarian, B., 1993. Multiple job scheduling with artificial neural networks. *Computers and Electrical Engineering*, **19** (2), 87–101.
- Lu, C. C., Ying, K. C., and Lin, S. W., 2014. Robust single machine scheduling for minimizing total flow time in the presence of uncertain processing times. *Computers and Industrial Engineering*, **74** (1), 102–110.
- Martí, L., García, J., Berlanga, A., and Molina, J. M., 2011. Multi-objective optimization with an adaptive resonance theory-based estimation of distribution algorithm: A comparative study. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Vol. 6683 LNCS. Springer, Berlin, Heidelberg, pp. 458–472.
- McCulloch, W. S. and Pitts, W., 1943. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, **5** (4), 115–133.

Bibliography

- Mehta, A., 2019. A Comprehensive Guide to Types of Neural Networks. *Digitalvidya*, , 2.
URL <https://www.digitalvidya.com/blog/types-of-neural-networks/>
- Mendes, J. M., 2013. A two-step optimization approach for job shop scheduling problem using a genetic algorithm Instituto Superior de Engenharia do Porto. *J. Magalhães-Mendes*, **1**, 1–20.
- Miguelañez, E., Zalzalá, A. M. S., and Tabor, P., 2004. Automating the Analysis of Wafer Data Using Adaptive Resonance Theory Networks. In: *Adaptive Computing in Design and Manufacture VI*. Springer London, pp. 137–148.
- Morton, T. and Pentico, D., 1993. Heuristic scheduling systems with applications to production systems and project management.
- Nguyen, S., Mei, Y., and Zhang, M., 2017. Genetic programming for production scheduling: a survey with a unified framework. *Complex & Intelligent Systems*, **3** (1), 41–66.
- Ono, I., Yamamura, M., and Kobayashi, S., 1996. Genetic algorithm for job-shop scheduling problems using job-based order crossover. In: *Proceedings of the IEEE Conference on Evolutionary Computation*. IEEE, pp. 547–552.
- Palmes, P. P., Hayasaka, T., and Usui, S., 2005. Mutation-based genetic neural network. *IEEE Transactions on Neural Networks*, **16** (3), 587–600.
- Pan, Q.-K., Wang, L., and B, Q., 2008. A novel multi-objective particle swarm optimization algorithm for no-wait flow shop scheduling problems. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, **222** (4), 519–539.
- Pandya, A. S. and Macy, R. B., 1997. Pattern Recognition with Neural Networks in C ++ , by Abhijit S . Pandya , Robert. , **16**, 261–262.

- Pinedo, M. L., 2012. *Scheduling: Theory, Algorithms, and Systems*. Springer US.
- Qiang Gao, Keyu Qi, Yaguo Lei, and Zhengjia He, 2005. An Improved Genetic Algorithm and Its Application in Artificial Neural Network Training. In: 2005 5th International Conference on Information Communications & Signal Processing. IEEE, pp. 357–360.
- Qing-Dao-Er-Ji, R. and Wang, Y., 2012. A new hybrid genetic algorithm for job shop scheduling problem. *Computers and Operations Research*, **39** (10), 2291–2299.
- Rahmani Hosseinabadi, A. A., Vahidi, J., Saemi, B., Sangaiah, A. K., and Elhoseny, M., 2019. Extended Genetic Algorithm for solving open-shop scheduling problem. *Soft Computing*, **23** (13), 5099–5116.
- Ramanan, T. R., Sridharan, R., Shashikant, K. S., and Haq, A. N., 2011. An artificial neural network based heuristic for flow shop scheduling problems. *Journal of Intelligent Manufacturing*, **22** (2), 279–288.
- Rezaeian, J., Javadian, N., Tavakkoli-Moghaddam, R., and Jolai, F., 2011. A hybrid approach based on the genetic algorithm and neural network to design an incremental cellular manufacturing system. *Applied Soft Computing Journal*, **11** (6), 4195–4202.
- Ruiz, R., Şerifoğlu, F. S., and Urlings, T., 2008. Modeling realistic hybrid flexible flowshop scheduling problems. *Computers and Operations Research*, **35** (4), 1151–1175.
- Ruiz, R. and Maroto, C., 2006. A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research*, **169** (3), 781–800.

- Sabuncuoglu, I., 1998. Scheduling with neural networks: A review of the literature and new research directions. *Production Planning and Control*, **9** (1), 2–12.
- Sadeghi, J., Sadeghi, S., and Niaki, S. T. A., 2014. Optimizing a hybrid vendor-managed inventory and transportation problem with fuzzy demand: An improved particle swarm optimization algorithm. *Information Sciences*, **272**, 126–144.
- Satake, T., Morikawa, K., and Nakamura, N., 1994. Neural network approach for minimizing the makespan of the general job-shop. *International Journal of Production Economics*, **33** (1-3), 67–74.
- Seidgar, H., Zandieh, M., and Mahdavi, I., 2016. Bi-objective optimization for integrating production and preventive maintenance scheduling in two-stage assembly flow shop problem. *Journal of Industrial and Production Engineering*, **33** (6), 404–425.
- Senties, O. B., Azzaro-Pantel, C., Pibouleau, L., and Domenech, S., 2009. A neural network and a genetic algorithm for multiobjective scheduling of semiconductor manufacturing plants. *Industrial and Engineering Chemistry Research*, **48** (21), 9546–9555.
- Sivanandam, S. and Deepa, S., 2007. Evolutionary Computation. In: Introduction to Genetic Algorithms. Springer Berlin Heidelberg, pp. 1–13.
URL http://link.springer.com/10.1007/978-3-540-73190-0_{_}1
- Sivapathasekaran, C., Mukherjee, S., Ray, A., Gupta, A., and Sen, R., 2010. Artificial neural network modeling and genetic algorithm based medium optimization for the improved production of marine biosurfactant. *Bioresource Technology*, **101** (8), 2884–2887.

Bibliography

- Smith, S. D. and Escobedo, R. A., 1994. Engineering and manufacturing applications of ART-1 neural networks. In: IEEE International Conference on Neural Networks - Conference Proceedings. Vol. 6. IEEE, pp. 3780–3785.
- Smith, W. E., 1956. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, **3** (1-2), 59–66.
- Tahsien, S. M., Karimipour, H., and Spachos, P., 2020. Machine learning based solutions for security of Internet of Things (IoT): A survey. *Journal of Network and Computer Applications*, **161**, 102630.
- Talbi, E. G., 2009.
- Tang, L-X and Wu, Y. P., 2002. Genetic descent algorithm for hybrid flow shop scheduling. *Journal of Automation*, **28** (4), 637–641.
- Taskin, Z., 2019. Optimization vs. heuristics: Which is the right approach for your business?
- Team, G. L., 2020. Types of Neural Networks and Definition of Neural Network. URL <https://www.mygreatlearning.com/blog/types-of-neural-networks/https://www.mygreatlearning.com/blog/types-of-neural-networks/{#}typesofneuralnetworks>
- Tscherepanow, M., 2010. TopoART: A topology learning hierarchical ART network. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Vol. 6354 LNCS. Springer, Berlin, Heidelberg, pp. 157–167.
- Williamson, J. R., 1996. Gaussian ARTMAP: A neural network for fast incremental learning of noisy multidimensional maps. *Neural Networks*, **9** (5), 881–897.
- Wu, C. C., Hsu, P. H., Chen, J. C., and Wang, N. S., 2011. Genetic algorithm for minimizing the total weighted completion time scheduling problem with

- learning and release times. *Computers and Operations Research*, **38** (7), 1025–1034.
- Wu, C. C., Liu, S. C., Zhao, C., Wang, S. Z., and Lin, W. C., 2018. A Multi-Machine Order Scheduling with Learning Using the Genetic Algorithm and Particle Swarm Optimization. *Computer Journal*, **61** (1), 14–31.
- Wu, W. H., Wu, W. H., Chen, J. C., Lin, W. C., Wu, J., and Wu, C. C., 2015. A heuristic-based genetic algorithm for the two-machine flowshop scheduling with learning consideration. *Journal of Manufacturing Systems*, **35**, 223–233.
- Yang, C., Ge, S. S., Xiang, C., Chai, T., and Lee, T. H., 2008. Output feedback NN control for two classes of discrete-time systems with unknown control directions in a unified approach. *IEEE Transactions on Neural Networks*, **19** (11), 1873–1886.
- Yen, B. P. and Wan, G., 2003. Single machine bicriteria scheduling: A survey. , **10** (3), 222–231.
- Yilmaz Eroglu, O. H., D. and K˘oksal, S., 2014. A GENETIC ALGORITHM FOR THE UNRELATED PARALLEL MACHINE SCHEDULING PROBLEM WITH JOB SPLITTING AND SEQUENCE-DEPENDENT SETUP TIMES - LOOM SCHEDULING. *Tekstil ve Konfeksiyon*, **24** (1), 66–73.
- Yu, H. and Liang, W., 2001. Neural network and genetic algorithm-based hybrid approach to expanded job-shop scheduling. *Computers and Industrial Engineering*, **39** (3-4), 337–356.
- Zhang, G., Gao, L., and Shi, Y., 2011. An effective genetic algorithm for the flexible job-shop scheduling problem. *Expert Systems with Applications*, **38** (4), 3563–3573.

Bibliography

Zhang, J., Ding, G., Zou, Y., Qin, S., and Fu, J., 2019. Review of job shop scheduling research and its new perspectives under Industry 4.0. *Journal of Intelligent Manufacturing*, **30** (4), 1809–1830.