Job Scheduling Problem Using Rough Fuzzy Multilayer Perception Neural Networks

Arindam Chaudhuri*, Kajal De**

*Lecturer (Computer Science Engineering), Birla Institute of Technology, Mesra, Patna Campus, Patna, India.
Email: arindam_chau@yahoo.co.in

**Professor in Mathematics, School of Science, Netaji Subhas Open University, Kolkata, India

Submitted: 01/01/2010
Accepted: 06/02/2010
Appeared: 16/02/2010

Abstract—Job Scheduling is a decision making process where next operation is selected to a partial schedule from set of competing operations with objective of minimizing performance measure. A complete schedule is consequence of best selected decisions. However, there exists an inherent degree of uncertainty in such problems. Here, we develop Rough Fuzzy Multi Layer Perception Neural Network Scheduler for Job Scheduling Problem. Genetic Algorithms generate optimal schedules to known benchmark problem. In each optimal solution, every individually selected operation of job is treated as decision which contains knowledge. Each decision is function of job characteristics divided into classes using domain knowledge. The Scheduler enhances classification strength and captures predictive knowledge regarding assignment of operation’s position in sequence. The trained network successfully replicates the performance of Genetic Algorithms. The better performance of Scheduler on test problems demonstrates the utility of method. The scalability of Scheduler on large problem sets gives satisfactory results.

Keywords: Decision Making, Genetic Algorithms, Job Scheduling, Rough Fuzzy Multi Layer Perception Neural Network.

1. INTRODUCTION

Scheduling involves sequencing of activities under time and resource constrains to meet a specific objective (Baker, 1974; French, 1982; Leung, 2004; Pinedo, 2005). It is a complex decision making problem because of conflicting goals, limited resources and the difficulty in accurately modeling real world problems (Sule, 2007). In an industrial job assignment problem context, scheduling activities are mapped to operations, and resources to machines. The purpose of scheduler is to determine starting time for each operation to achieve desired performance measures, while satisfying capacity and technological constraints. In today’s highly competitive industrial environment, there is an urgent need for robust and flexible approach capable of generating good solutions within an acceptable timeframe (Agarwal et. al., 2006; Alagoz et. al., 2003; Allahverdi et. al., 2002; Bestwick et. al., 1976; Blackstone et. al., 1982; Brown et. al., 1966; Chandrasekaran et. al., 2005; Fonseca et. al., 2002; Foo et. al., 1995; Grabowski et. al., 2007; Gupta et. al., 1991; Käschel et. al., 1999; Laha et. al., 2007; Lawrence, 1984; Rabelo et. al., 1989; Ruiz et. al., 2005, 2007; Schuster et. al., 2003; Sethanun et. al., 2004; Shah et. al., 2004; Soewandi 1998; Wang et. al., 2003). Scheduling theory is concerned with formulation and study of various scheduling models and development of associated solution techniques (Wiers, 1997). Some widely studied classical models are: single machine, parallel machine, flow scheduling and job scheduling models (Baker, 1974; Chandrasekaran et. al., 2005; Fink et. al., 2003). Of these, deterministic job scheduling model has attracted most attention for two reasons viz., (i) The generic formulation of model makes it applicable to possibly all scheduling domains; (ii) The problem’s intractable nature has inspired researchers to develop broad spectrum of strategies, ranging from simple heuristics to adaptive search strategies based on conceptual framework borrowed from Biology, Genetics and Evolution (Dietterich, 1996; Goldberg, 1989; Mitchell, 1997). The deterministic Job Scheduling Problem (JSP) consists of finite set of jobs to be processed on finite set of machines. The difficulty in JSP lies in number of possible schedules. Generally, for \( n \times m \) JSP, the cardinality of set of possible schedules is \((n!)^m\). Though the set of feasible schedules in which precedence constraints have not been violated is subset of this set, it is still large enough to discourage complete enumeration for even moderately sized problems. The computation time for algorithms searching possible solution space to identify an optimal schedule increases exponentially with problem size. Hence, JSP belongs to the set of problems classified as NP hard Problems (French, 1982; Giffler et. al., 1960; Wiers, 1997).
The search based approach to JSP is based on exploration of feasible solution space to identify an optimal solution. Adaptive search algorithms like Genetic Algorithms (GA), Tabu Search and Simulated Annealing have been applied to this domain with success and in many cases are capable of providing optimal or near optimal solutions. Heuristics offer knowledge based alternative to the problem (Wiers, 1997). Many dispatching rules are abstractions formulated from expert knowledge of the problem. Elementary priority dispatch rules such as Shortest Processing Time (SPT), First Come First Served (FCFS) and Earliest Due Date (EDD) etc. have proven useful in simulation studies of job environment (Blackstone et. al., 1982; Käschel et. al., 1999). The ease of application, rapidity of computation and flexibility to changing shop floor conditions are key reasons for heuristic based approaches to be used widely in many industrial situations. Their main weakness however is that different heuristics cater to different problems and no single heuristic dominates rest across all situations. A key shortcoming of adaptive search methods for scheduling problem is the lack of insight offered by them into their decision making process. The stochastic nature of search in these algorithms yield good solutions but does not help explain the process by which they are obtained or properties attributable to provided solutions. An interesting line of investigation would be to cast the scheduling problem as learning task with the goal of capturing properties of known good solutions. In such a formulation, optimal solutions generated by efficient optimizers provide desired learning objects. In these optimized sequences, each individual operation is treated as a decision which captures some problem specific knowledge. Hence, these solutions contain valuable information such as relationship between an operation’s attributes and its position in sequence (solution). An exploration of these sequences by Machine Learning techniques would capture predictive knowledge regarding assignment of operation’s position in a sequence based on its attributes. While this approach learns knowledge contained in schedules produced by an optimization method, there is no reason that knowledge contained in schedules produced by human scheduler could not serve as training set. In this way, training Artificial Neural Network (ANN) scheduler can serve as a mechanism for capturing knowledge contained in historical scheduling decisions and be an invaluable aid in Industrial scheduling domain (Cheung, 1994; Fonseca et. al., 2002; Foo et. al., 1995; Jain et. al., 1998; Principe et. al., 2000; Widrow et. al., 1994; Yang et. al., 2000).

The major prima facie of this work is to use Rough Fuzzy Multi Layer Perceptron Neural Network (RFMLP-NN) as hybridized Machine Learning tool (Pal et. al., 1999) to study scheduling process for JSP. RFMLP-NN is basically a Soft Computing paradigm in which consortium of methodologies works synergistically and provides flexible information processing capability for handling real life ambiguous situations (Zadeh, 1994). The main ingredients of RFMLP-NN are Rough Sets, Fuzzy Logic and Multi Layer Perceptron ANN (Han et. al., 1999). As job scheduling is basically a decision making process, it is obvious that there exist an inherent degree of uncertainty, vagueness and imprecision associated with such problems. These aspects are taken care of by using Fuzzy Sets and Rough Sets (Pawlak, 1982, 1991; Zadeh, 1965). The Fuzzy Sets helps in handling vagueness and imprecision in linguistic input description and ambiguity in output decision. It generalizes classical two-valued logic for reasoning under uncertainty. To achieve this, notation of membership in a set needs to become a matter of degree. Two things are accomplished by doing this viz., (i) ease of describing human knowledge involving vague concepts and (ii) enhanced ability to develop cost-effective solution to real-world problem. It is thus a multi-valued logic which is model-less approach and clever disguise of Probability Theory (Zadeh, 1965). The Rough Sets deals with uncertainty arising from inexact or incomplete information and extracts the crude domain knowledge for determining network parameters. It synthesizes approximation of concepts and finds hidden patterns in the acquired data. It aids in representation and processing of both qualitative and quantitative parameters in reduced form and mixes user defined and measured data thus evaluating significance of data. It classifies decision rules from data and provides legible and straightforward interpretation of synthesized models. It is mostly suited for parallel processing applications and hence can be effectively integrated with ANN (Pawlak, 1982, 1991). ANN is data-driven modeling tool that is able to capture and represent complex and non-linear input/output relationships. They are recognized as powerful and general technique for Machine Learning because of their non-linear modeling abilities and robustness in handling noise-ridden data (Dietterich, 1996; Mitchell, 1997; Principe et. al., 2000). They are composed of several layers of processing elements or nodes. These nodes are linked by connections with each connection having an associated weight. The weight of connection is measure of its strength and its sign is indicative of excitation or inhibition potential. It captures task relevant knowledge as part of its training regimen. This knowledge is encoded in network in architecture or topology of network. The transfer functions are used for nonlinear mapping along with set of network parameters (weight and biases). To generate learning, knowledge base GA are chosen for producing optimal solutions as they have proved to be successful in empirical scheduling research (Goldberg, 1989). The job of RFMLP-NN is thus to capture predictive knowledge regarding assignment of operation’s position in job sequence. The organization of this Paper is as follows. In section 2, a brief discussion on work related to JSP is given. The next section discusses some preliminaries of Rough Sets relevant to this work. The section 4 presents methodology of RFMLP-NN model for JSP. In section 5, results are discussed. Finally, in section 6 conclusions are given.

2. RELATED WORK ON JOB SCHEDULING PROBLEM

Artificial Intelligence (AI) aims at developing machines and programs that have capability to learn, adapt and exhibit human
like intelligence (Dietterich, 1996; Mitchell, 1997; Principe et. al., 2000; Tuban et. al., 2001; Zadeh, 1994). Hence, learning algorithms are important for practical applications of AI. The field of Machine Learning is the study of methods for programming computers to learn (Dietterich, 1996). Many important algorithms have been developed and successfully applied to diverse learning tasks such as Speech Recognition, Game Playing, Medical Diagnosis, Financial Forecasting and Industrial Control (Mitchell, 1997). There have been several applications of ANN in scheduling (Agarwal et. al., 2006; Alagöz et. al., 2003; Cheung, 1994; Dagli et. al., 1995; Fisher et. al., 1963; Foo et. al., 1988; Jain et. al., 1996, Kumar et. al., 2005; Laha et. al., 2007; Lin et. al., 2004; Luangpaiboon et. al., 2004; Rabelo et. al., 1989, Yu et. al., 2001). A comprehensive survey of ANN architectures used in scheduling (Cheung, 1994). These are basically searching networks (hopfield networks), probabilistic networks (boltzmann machine), error correcting networks (multi layer perception), competing networks and self organizing networks. An investigation and review of application of ANN in JSP provided by (Jain et. al., 1996). ANN has been used in many important applications such as Function Approximation, Pattern Recognition and Classification, Memory Recall, Prediction, Optimization and Noise Filtering. Many commercial products such as Modems, Image Processing and Recognition Systems, Speech Recognition Software, Data Mining, Knowledge Acquisition Systems and Medical Instrumentation etc. have been developed using ANN (Widrow et. al., 1994). ANN was first employed to JSP (Foo et. al., 1988). They formulated the scheduling problem as an integer liner programming problem and used a modified hopfield network to model the problem. The energy function for network is linear function and is the sum of starting times of jobs. In a later work, (Foo et. al., 1995) investigated scaling properties of modified hopfield network for scheduling problem. Some of the drawbacks of these networks include lack of convergence, convergence to local minima and number of hardware processing elements. Adaptive ANN for generalized JSP was proposed by (Yang et. al., 2000) where precedence and resource constraints of the problem are mapped to network architecture. The network consists of linear segmented activation functions. The network generates feasible solutions, which are further improved by heuristics to obtain non-delay solutions. The other prominent ANN systems used in scheduling are error correcting networks which adapt network parameters (weights and biases) based on propagation of error between desired and computed output of network. A major class in such systems is MLP network, where supervised learning takes place by back propagation algorithm. A modified MLP model was proposed by (Jain et. al., 1996), where ANN performs task of optimization and outputs desired sequence. A novel input/output representation scheme is used to encode JSP for ANN training. Although the method has been able to handle large problem sizes (30 × 10) compared to other approaches, generalization capability of model is limited to approximately 20% deviation from training sample.

In contrast to above approach, many applications of error correcting networks to JSP utilize ANN as component of hybrid scheduling system. ANN was used to rank and determine coefficients of priority rules (Rabelo et. al., 1989). An Expert System utilizes these coefficients to generate schedules. GA was used for optimization and ANN performs multi-objective schedule evaluation (Dagli et. al., 1995). The network maps set of scheduling criteria to appropriate values provided by experienced schedulers. A hybrid approach was presented by (Yu et. al., 2001) for JSP in which GA was used for optimization of job sequences and ANN performs optimization of operation start times. This approach has been successfully tested on large number of simulation cases and practical applications. The use of MLP ANN for simulation of job environment was investigated by (Fonseca et. al., 2002). This work compares performance of ANN in estimating manufacturing lead time to traditional simulation approaches. In an approach very similar to ANN learning mechanism (Agarwal et. al., 2006) proposed an adaptive learning approach for flow shop scheduling problem. In their approach, heuristics were guided through problem search space based on weighted processing time. The weights were adaptive with two parameters, learning rate and reinforcement rate used to influence extent and memory of search. The authors report satisfactory performance on several sets of benchmark problems drawn from literature.

3. ROUGH SETS

This section presents some preliminaries of Rough Sets (Nguyen et. al., 1997; Pawlak, 1982, 1991; Peters et. al., 2000; Skowron et. al., 1992). An information system is defined by pair $S = \langle U, \mathcal{A}\rangle$, where $U$ is nonempty finite set called universe and $\mathcal{A}$ is nonempty finite set of attributes. An attribute $a$ can be regarded as function from domain $U$ to some value set $V_a$. A decision system is any information system of form $A = (U, \mathcal{A} \cup \{d\})$, where $d \notin \mathcal{A}$ is decision attribute. The elements of $\mathcal{A}$ are called conditional attributes. An information system can be represented as an attribute-value table, in which rows are labeled by objects of universe and columns by attributes. Similarly, decision system can be represented by decision table. With every subset of attributes $B \subseteq \mathcal{A}$, an equivalence relation $I_B$ can easily be associated on $U$, $I_B = \{(x, y) \in U: \forall a \in B, a(x) = a(y)\}$. Then, $I_B = \bigcap_{a \in B} I_a$. If $X \subseteq U$, sets $\{x \in U: [x]_B \subseteq X\}$ and $\{x \in U: [x]_B \cap X = \emptyset\}$, where, $[x]_B$ denotes equivalence class of object $x \in U$ relative to $I_B$, are called $B$-lower and $B$-upper approximation of $X$ in $S$ and denoted by $B_X$, $\overline{B}_X$ respectively. $X \subseteq U$ is $B$-exact or $B$-definable in $S$ if $B_X = \overline{B}_X$. It may be observed that $B_X$ is greatest $B$-definable set contained in $X$ and $\overline{B}_X$ is smallest $B$-definable set containing $X$. 

6
Now we define notions relevant to knowledge reduction. The aim is to obtain irreducible but essential parts of knowledge encoded by given information system, which constitutes reducts of the system. This in effect reduces to looking for maximal sets of attributes taken from initial set \( \Lambda \) which induce same partition on domain as \( \Lambda \). In other words, the essence of information remains intact and superfluous attributes are removed. Reducts have already been characterized by discernibility matrices and discernibility functions (Skowron et al., 1992). Consider \( U = \{x_1, \ldots, x_n\} \) and \( A = \{a_1, \ldots, a_m\} \) in information system \( S = \langle U, A \rangle \). By discernibility matrix \( M(S) \) of \( S \) is meant an \( n \times n \) matrix such that,

\[
c_{ij} = \{a \in A : a(x_i) \neq a(x_j)\} \tag{1}
\]

A discernibility function \( f_c \) is function of \( m \) Boolean variables \( \bar{a}_1, \ldots, \bar{a}_m \) corresponding to attributes \( a_1, \ldots, a_m \) respectively and is defined as follows:

\[
f_c(\bar{a}_1, \ldots, \bar{a}_m) = \bigwedge_{1 \leq i, j \leq n, j < i} \neg c_{ij} \tag{2}
\]

where, \( \bigwedge(c_{ij}) \) is disjunction of all variables \( \bar{a} \) with \( a \in c_{ij} \). It is observed that \( \{a_1, \ldots, a_m\} \) is reduct in \( S \) if and only if \( a_1 \wedge \ldots \wedge a_m \) is prime implicant of \( f_c \).

4. ROUGH FUZZY MULTI LAYER PERCEPTION NEURAL NETWORK MODEL FOR JOB SCHEDULING PROBLEM

The objective of this work is to develop RFMLP-NN scheduler (Peters et al., 2000). To accomplish this, learning task for ANN needs to be identified. The central issue of this investigation is that optimal solutions to scheduling problem have common features which can be implicitly captured by Machine Learning tool like ANN. The second issue is that learned function i.e., ANN can be utilized to generate good solutions to new problems. The third issue concentrates in handling inherent uncertainty and imprecision involved in scheduling problems which is taken care of by Rough and Fuzzy Sets. To generate learning or knowledge base GA are chosen for producing optimal solutions as they have proved to be successful in empirical scheduling research. Scheduling can be understood as decision making process (Tuban et al., 2001), where decision is selection of next operation to add to a partial schedule from among set of competing operations with objective of minimizing chosen performance measure. A complete schedule is thus consequence of repeated decisions to select best operation. Thus, in an optimal solution every individually scheduled operation of job is treated as decision which contains knowledge. Each decision is modeled as function of set of job characteristics such as processing time, machine load, which are divided into classes using domain knowledge from common dispatching rules like shortest processing time. The job of RFMLP-NN is then to capture predictive knowledge regarding assignment of operation’s position in the sequence. A schematic illustration of entire procedure is given in figure 1. The first task is to generate optimal solutions to benchmark scheduling problem using GA. The second task is to model scheduling function as Machine Learning problem by defining classification scheme, which maps optimal sequences to data patterns. The third task is to develop an appropriate ANN model incorporating concepts of Rough Sets and Fuzzy Sets. ANN is trained on classification data patterns. The final task focuses on development of scheduler which combines well known active scheduling algorithm with ANN to obtain schedules for new problems. The different tasks are explained in following subsections.

4.1 Generation of solutions using Genetic Algorithms

The knowledge base for learning task is constructed from optimal solutions to JSP generated by GA (Yamada et al., 1992, 1997). A famous \( 6 \times 6 \) problem instance viz., ft06 devised by (Fisher et al., 1963), has been chosen as benchmark problem. In this regard, makespan of each scheduling operation is considered as performance measure (Rajendran et al., 2004; Soewandi, 1998). This instance has six jobs, each with six operations to be scheduled on six machines and has known optimum makespan of 55 units. The data for instance is shown in Table 1 using the structure (machine, processing time). A distributed GA implemented in Java developed by (Shah et al., 2004), is utilized in this work for obtaining solutions to benchmark problem. A solution generated by GA is sequence of 36 operations (6 jobs \( \times \) 6 machines): \{1, 3, 2, 4, 6, 2, 3, 4, 3, 6, 6, 2, 5, 5, 3, 5, 1, 1, 6, 4, 4, 4, 1, 2, 5, 3, 2, 3, 6, 1, 5, 2, 1, 6, 4, 5\}. Each number in sequence is representative of job number and current operation number. The repetition of job number in sequence indicates next operation of that job. The representation of GA solution is shown in figure 2. On benchmark instance shown in table 1, GA is run 3000 times. Each run produced solution and optimal makespan of 55 units was achieved 1696 times. The next step is to transform these 1696 solutions or chromosomes into data structure suitable for classification task.

4.2 Data Classification Problem

The solutions obtained by GA contain valuable information relevant to scheduling process (Goldberg, 1989; Koonce, 2000). The learning task is to predict position of an operation in GA sequence denoted by chromosome, based on its features or attributes. A sequence contains information about ordering of operations on each machine. A schedule specifies both sequence and starting times of operations. The reason for utilizing sequences produced by GA instead of schedules for learning task is twofold:

1) In GA, chromosomes represent solutions to Optimization Problem. In constructing GA for scheduling, decision to represent chromosomes either as sequences or schedules is design decision. In former case, decoder is specified to map sequence to schedule. The genetic operators needed to
manipulate sequences are simple than those needed to manipulate schedules. The efficiency of GA for complicated combinatorial Optimization Problem like JSP is an important aspect and hence sequences are utilized to represent chromosomes.

2) There exists an N:1 mapping between sequences and schedules. This implies that in operation in different positions in two sequences might still occupy same position in decoded schedule. As prediction of position of operation is objective of learning task, sequences were utilized instead of schedules to prevent any loss of information relevant to learning task.

Based on study of operation attributes commonly used in priority dispatch rules, attributes that have been identified as input features are operation, process time, remaining time and machine load. These input features have been clustered into different classes using concept hierarchy for JSP (Koonce, 2000). Each job in benchmark problem has six operations that must be processed in given sequence. The operation feature identifies sequence number of operation ranging between 1 and 6. This feature has been clustered into four classes as: {1} first, {2, 3} middle, {4, 5} later and {6} last. The process time feature represents processing time for operation. The remaining time feature denotes sum of processing times for remaining operations of that job and provides measure of work remaining to be done for completion of job assuming no resource conflicts. For benchmark problem viz., f06 instance, processing time ranges from 1 to 10 units, while remaining times ranged from 0 to 39 units. Based on data, three classes or clusters for these features are identified with ranges being split into three equal intervals and classified as short, medium and long. The machine load feature determines machine loading and is clustered into two classes’ viz., light and heavy. This feature represents capacity or utilization of machines in units of time and helps in differentiating between possible bottleneck and non-bottleneck machines (Adams et. al., 1988). These input features thus signify features of operation that affect its placement in schedule. The target concept to be learned is priority or position in sequence. Since an operation can be positioned in any one of 36 locations available in GA sequence, it may be difficult to discover an exact relationship between input features and position. However, if problem is modified to predict range of locations for operation, learning task becomes simpler. The target feature priority, thus determines range of positions in sequence where operation can be inserted. The possible range of positions have been split into 6 classes and assigned class labels as shown in table 2. The classification problem with input and the target features is illustrated in figure 3.

![Fig. 1. Schematic illustration of RFMLP-NN based Scheduler](image-url)
Various important aspects should be considered for developing an effective ANN model. We consider two such aspects. The first criterion is selection of suitable architecture, training algorithm, learning constants and termination issues for building the model. The second criteria relate to determination of sizes and choice of data patterns for training, cross validation and testing data sets. Considerable experimentation is required to achieve good model of the data. ANN model chosen for data classification is MLP ANN. An MLP consists of group of nodes arranged in layers. Each node in a layer is connected to all nodes in next layer by links which have weight associated with them. The input layer contains nodes that represent input features of classification problem. A real valued feature is represented by single node, whereas discrete feature with n distinct values is represented by n input nodes. The classification strength of MLP ANN is enhanced by incorporating Rough Sets and Fuzzy Sets in ANN (Pal et al., 1986, 1999), which results in hybridization of connectionist paradigm for Pattern Classification. ANN acts as efficient connectionist between the two. In this hybridization, Fuzzy Sets help in handling linguistic input information and ambiguity in output decision, while Rough Sets extract domain knowledge for determining network parameters. Now, we discuss important features of RFMLP-NN. A brief discussion of Fuzzy MLP is given first and then concept of Rough Sets is incorporated in network.

The Fuzzy MLP model incorporates fuzziness at input and output levels of MLP as shown in figure 3 and is capable of handling exact or numerical and inexact or linguistic forms of input data (Pal et al., 1986). Any input feature value is described in terms of some combination of membership values in linguistic property sets low (L), medium (M) and high (H). Class membership values \( \mu \) of patterns are represented at output layer of Fuzzy MLP. During training, weights are updated by back propagation errors with respect to these membership values such that contribution of uncertain vectors is automatically reduced. A four layered feed forward MLP is used here. The output of neuron in any layer \( h \) other than input layer (\( h = 0 \)), is given as:

\[
y^{(h)}_j = \frac{1}{1 + \exp(-\sum_{i} y^{(h-1)}_{j} w_{ji}^{(h-1)})} \quad (3)
\]

where, \( y^{(h-1)}_i \) is state of \( i^{th} \) neuron in preceding \((h-1)^{th}\) layer and \( w_{ji}^{(h-1)} \) is weight of connection from \( i^{th} \) neuron in layer \((h-1)\) to \( j^{th} \) neuron in layer \( h \). For nodes in input layer, \( y^{(0)}_j \) corresponds to \( j^{th} \) component of input vector. Further it is to be noted that \( x^{(h)}_j = \sum_{i} y^{(h-1)}_{ji} w_{ji}^{(h-1)} \).

### 4.3 Rough Fuzzy Multi Layer Perception Neural Network Model

An \( n \)-dimensional pattern \( F_i = [F_{i1}, F_{i2}, \ldots, F_{in}] \) is represented as a \( 3n \)-dimensional vector,

\[
F_i = [\mu^{low}_{i1}(F_i), \ldots, \mu^{high}_{i1}(F_i), \mu^{low}_{i2}(F_i), \ldots, \mu^{high}_{i2}(F_i), \ldots, \mu^{low}_{in}(F_i), \ldots, \mu^{high}_{in}(F_i)] \quad (4)
\]

where, \( \mu \) values indicate membership functions of corresponding linguistic \( \pi \)-sets low, medium, and high along each feature axis and \( y^{(0)}_1, y^{(0)}_2, \ldots, y^{(0)}_n \) refer to activations of \( 3n \) neurons in input layer. When input feature is exact in nature, \( \pi \)-Fuzzy sets in one dimensional form are used with range \([0, 1]\) and are represented as:

### Table 1. The ft06 problem instance devised by (Fisher and Thompson, 1963)

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.3</td>
<td>1.3</td>
<td>2.6</td>
<td>4.7</td>
<td>6.3</td>
<td>5.6</td>
</tr>
<tr>
<td>2</td>
<td>2.5</td>
<td>3.5</td>
<td>5.5</td>
<td>6.1</td>
<td>1.0</td>
<td>4.4</td>
</tr>
<tr>
<td>3</td>
<td>3.5</td>
<td>4.4</td>
<td>6.9</td>
<td>1.9</td>
<td>2.1</td>
<td>5.7</td>
</tr>
<tr>
<td>4</td>
<td>2.5</td>
<td>1.5</td>
<td>3.5</td>
<td>4.3</td>
<td>5.8</td>
<td>6.9</td>
</tr>
<tr>
<td>5</td>
<td>3.9</td>
<td>2.3</td>
<td>5.5</td>
<td>6.4</td>
<td>1.3</td>
<td>4.1</td>
</tr>
<tr>
<td>6</td>
<td>2.3</td>
<td>4.3</td>
<td>6.9</td>
<td>1.1</td>
<td>10.5</td>
<td>3.1</td>
</tr>
</tbody>
</table>

### Table 2. Assignment of class labels to target feature

<table>
<thead>
<tr>
<th>Range of Positions</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 6</td>
<td>Zero</td>
</tr>
<tr>
<td>7 – 12</td>
<td>One</td>
</tr>
<tr>
<td>13 – 18</td>
<td>Two</td>
</tr>
<tr>
<td>19 – 24</td>
<td>Three</td>
</tr>
<tr>
<td>25 – 30</td>
<td>Four</td>
</tr>
<tr>
<td>31 – 36</td>
<td>Five</td>
</tr>
</tbody>
</table>
where, $\lambda > 0$ is radius of $\pi$-function with $c$ as central point.

When input feature $F_j$ is linguistic in nature, its membership values for $\pi$-sets low L, medium M and high H are quantified as:

\[
\begin{align*}
\pi(F_j; c, \lambda) &= \begin{cases} 
2(1 - \frac{F_j - c}{\lambda}), & \text{for } \frac{\lambda}{2} \leq F_j - c \leq \lambda \\
1 - 2(1 - \frac{F_j - c}{\lambda}), & \text{for } 0 \leq F_j - c \leq \frac{\lambda}{2} \\
0, & \text{otherwise}
\end{cases}
\end{align*}
\] (5)

Now we consider the procedure for selecting centers and radii of overlapping $\pi$-sets. Let $m_j$ be mean of pattern points along $j$th axis. Then, $m_{jL}$ and $m_{jH}$ are defined as mean along $j$th axis of pattern points having co-ordinate values in the range $[F_{jmL}, m_j]$ and $(m_j, F_{jmH})$ respectively, where $F_{jmL}$ and $F_{jmH}$ denote upper and lower bounds of dynamic range of feature $F_j$ for training set considering exact values only. For three linguistic property sets, centers and corresponding radii are defined as:

\[
c_{\text{medium}}(F_j) = m_{jM};
\]

\[
c_{\text{low}}(F_j) = m_{jL};
\]

\[
c_{\text{high}}(F_j) = m_{jH};
\]

\[
\lambda_{\text{low}}(F_j) = 2(c_{\text{medium}}(F_j) - c_{\text{low}}(F_j));
\]

\[
\lambda_{\text{high}}(F_j) = 2(c_{\text{high}}(F_j) - c_{\text{medium}}(F_j));
\]

\[
\lambda_{\text{middle}}(F_j) = \frac{(c_{\text{middle}}(F_j) - c_{\text{low}}(F_j)) \times (c_{\text{medium}}(F_j) - c_{\text{high}}(F_j))}{F_{jmL} - F_{jmH}}.
\] (8)

Here, we take into account distribution of pattern points along each feature axis while choosing corresponding centers and radii of linguistic properties. Besides, the amount of overlap between three linguistic properties can be different along different axis depending on pattern set.

**CLASSIFICATION PROBLEM**

![Fig. 3.Input features and target class for Classification Problem](image)

**4.3.2 Output Representation**

Let us consider an $l$-class problem domain such that we have $l$ nodes in output layer. Also consider $n$-dimensional vectors $\alpha_k = [\alpha_{k1}, ..., \alpha_kl]$ and $\nu_k = [\nu_{k1}, ..., \nu_kl]$ denote mean and standard deviation respectively, of exact training data for $k$th class $c_k$. The weighted distance of training pattern $F_i$ from $k$th class $c_k$ is defined as:

$$d_k = \sqrt{\sum_{j=1}^{l} \left( \frac{F_{ij} - \alpha_{kj}}{\nu_{kj}} \right)^2}.$$
where, \(F_{ij}\) is value of \(j^{th}\) component of \(i^{th}\) pattern point. The membership of \(j^{th}\) pattern in class \(k\), lying in the range [0, 1] is defined as:

\[
\mu_k(F_i) = \frac{1}{1 + \left(\frac{d_{ij}}{f_d}\right)^c}
\]

where, positive constants \(f_d\) and \(f_r\) are denominational and exponential fuzzy generators controlling amount of fuzziness in class membership set. Then, for \(i^{th}\) input pattern, desired output of \(f^k\) output node is defined as:

\[
d_j = \mu_j(F_i)
\]

According to this definition a pattern can simultaneously belong to more than one class, and this is determined from training set during learning phase.

4.3.3 Network configuration using Rough Sets

We formulate rule generation and knowledge encoding for configuring Fuzzy MLP ANN by using Rough Sets (Pal et al., 1999). The algorithm is able to deal with multiple objects corresponding to one decision attribute. From the perspective of Pattern Recognition, this implies using multiple prototypes to serve as representatives of any arbitrary decision region (Pal et al., 1986). The entire procedure is represented diagrammatically in figure 4.

4.3.4 Dependency Rule Generation

The principal task in rule generation is to compute reducts with respect to particular kind of information system and decision system (Pal et al., 1999; Pawlak, 1982, 1991; Peters et al., 2000). We use relativised versions of matrices and functions viz., \(d\)-reducts and \(d\)-discernibility matrices as basic tools for computation. The methodology is discussed below. Let \(S = (U, \mathcal{A})\) be a decision table, with \(C\) and \(D = \{d_1,\ldots,d_l\}\) its sets of condition and decision attributes respectively. We divide decision table \(S = (U, \mathcal{A})\) into \(l\) tables \(S_i = (U, A_i)\); \(i = 1,\ldots,l\), corresponding to \(l\) decision attributes \(d_i,\ldots,d_l\), where \(U = U_1 \cup \cdots \cup U_l\) and \(A_i = C \cup \{d_i\}\). Let \(\{x_1,\ldots,x_n\}\) be set of those objects of \(U\) that occur in \(S_i\); \(i = 1,\ldots,l\). Now for each \(d_i\)-reduct \(B = \{b_1,\ldots,b_k\}\), a discernibility matrix denoted by \(M_{d_i}(B)\) from \(d_i\)-discernibility matrix is defined as follows:

\[
c_{ij} = \{a \in B : a(x_i) \neq a(x_j)\}, \quad (12)
\]

For each object \(x_j \in \{x_1,\ldots,x_n\}\), discernibility function \(f_{d_j}^{x_j}\) is defined as:

\[
f_{d_j}^{x_j} = \Lambda(\mathcal{V}(c_{ij}) : 1 \leq i, j \leq n, j < i, c_{ij} \neq \Phi).
\]

where, \(\mathcal{V}(c_{ij})\) is disjunction of all members of \(c_{ij}\). Then, \(f_{d_j}^{x_j}\) is brought to its conjunctive normal form. Thus, dependency rule \(r_j\) is obtained, viz., \(P_i \leftarrow d_i\), where \(P_i\) is disjunctive normal form \(f_{d_j}^{x_j}\), \(j = i_1,\ldots,i_p\). The dependency factor \(d_{ij}\) for \(r_j\) is given as:

\[
d_{ij} = \frac{\text{card}(POS_{c_i}(d_j))}{\text{card}(U)}
\]

where, \(POS_{c_i}(d_j) = \bigcup_{x \in c_i} l(X)\) and \(l(X)\) is lower approximation of \(X\) with respect to \(l\). In this case, \(d_{ij} = 1\).

4.3.5 Knowledge Encoding

In knowledge encoding phase consider feature \(F_j\) for class \(c_k\) in \(l\)-class problem domain. The inputs for \(i^{th}\) representative sample \(F_i\) are mapped to corresponding three-dimensional feature space \(\mu_{\text{low}}(F_i), \mu_{\text{medium}}(F_i), \mu_{\text{high}}(F_i)\). Let these be represented by \(L_p, M_p, H_p\) respectively. As the method considers multiple objects in a class, separate \(n_k \times 3n\)-dimensional attribute-value decision table is generated for each class \(c_k\) where, \(n_k\) indicates number of objects in \(c_k\). The absolute distance between each pair of objects is computed along each attribute \(L_p, M_p, H_p\) for all \(j\). The equation (12) is modified to directly handle real-valued attribute table consisting of fuzzy membership values.

We define \(c_{ij} = \{a \in B : |a(x_i) - a(x_j)| \geq Th\}\), for \(i, j = 1,\ldots,n\) where, \(Th\) is an adaptive threshold. It is to be noted that adaptivity of this threshold is built in, depending on inherent shape of membership function.

While designing initial structure of RFMLP-NN, union of rules of \(l\) classes is considered. The input layer consists of \(3n\) attribute values while output layer is represented by \(l\) classes. The hidden layer nodes model innermost operator in antecedent part of rule, which can be either conjunct or disjunct. The output layer nodes model outer level operands, which can again be either conjunct or disjunct. For each inner level operator, corresponding to one output class one dependency rule, one hidden node is dedicated. Only those inputs attribute that appear in this conjunct or disjunct are connected to appropriate hidden node, which, in turn, is connected to corresponding output node. Each outer level operator is modeled at output layer by joining corresponding hidden nodes. It is to be noted that single attribute involving no inner level operators is directly connected to appropriate output node via hidden node to maintain uniformity in rule mapping.
4.4 Structure of Rough Fuzzy Multi Layer Perception Neural Network Model

Let us now design initial structure of four layered RFMLP-NN. The classification problem under consideration has four discrete input features or attributes with four, three, three and two distinct values. Thus, input layer in RFMLP-NN has a total of \((12 = 4 + 3 + 3 + 2)\) nodes, each representative of an input feature value. The hidden layers map input to output layer and number of nodes in hidden layers is empirically chosen to obtain a good model. The nodes in hidden layers model innermost operator in antecedent part of rule, which can either be conjunct or disjunct. The output layer gives decision of classifier which is represented by classes. Each output node represents a possible target class and hence there are six output nodes in RFMLP-NN. The nodes in this layer model outer level which can again be either conjunct or disjunct. The dependency factor considered for any rule is \(1\). Only those input attributes that appear in conjunct or disjunct are connected to appropriate hidden nodes, which in turn are connected to corresponding output nodes. The winner takes all heuristic is used to determine class membership when multiple nodes are present in output layer. The class of output node with maximum activation or output is class computed by network.

Next we proceed to the description of initial weight encoding procedure. Let dependency factor for particular dependency rule for class \(c_i\) be given by \(df = \alpha = 1\). The weight \(w_{kj}^{(i)}\) between hidden node \(i\) and output node \(k\) is equal to \(\frac{\alpha}{fac} + \epsilon\), where \(fac\) refers to number of outer level operands in antecedent of rule and \(\epsilon\) is small random number taken to destroy any symmetry among weights. Generally \(fac \geq 1\) and each hidden node get connected to only one output node. Let initial weight considered at hidden node is denoted as \(\beta\). The weight \(w_{aj}^{(i)}\) between an attribute \(a_j\) (where, the attribute \(a\) corresponds to low \(L\), medium \(M\), or high \(H\)) and hidden node \(i\) is equal to \(\frac{\beta}{facd} + \epsilon\), such that \(facd\) is number of attributes connected by corresponding inner level operator.

Again, we see that \(facd \geq 1\). Thus, for an \(i\)-class problem domain, there are at least \(i\) hidden nodes. It may be noted that number of hidden nodes is determined directly from dependency rules. It depends on form in which antecedents are present in rules. A training algorithm is also used to determine values of network parameters which best map a given set of input patterns to desired outputs. The most commonly used training algorithm is back propagation algorithm (Rumelhart et. al., 1986). The term back propagation refers to direction of propagation of error. The goal of training regimen is to adjust weights and biases of network to minimize chosen cost function (Jain et. al., 1996). Though several cost functions are available, function appropriate for classification problems is cross entropy function which is a measure of relative entropy between different classes. This function needs to be minimized and back propagation algorithm uses a form of gradient descent to update weights. A learning rate scales correction term in revising weights during training process and hence governs rate of learning. In addition, momentum parameter is also used to speed up training process. This parameter further scales correction based on memory of size of previous increment to weight. In RFMLP-NN model, we used a variant of back propagation algorithm with momentum learning (Principe et. al., 2000). The network is initialized with random weights and training algorithm modifies weights according to procedure discussed above. Since random initial weights were specified, network was trained multiple times before best model was chosen. Further, in each run entire training data set is presented to network multiple times and each view is known as an epoch. The number of epochs per run and number of runs are user specified inputs to training algorithm. Thus, 1696 optimal schedules obtained by GA represented total of 61056 scheduled operations (1696 schedules \(\times 36\) operations/schedule). Assignment of input features and target classes was done for each operation according to classification scheme described in previous subsection. Sample data for classification task is shown in table 3. The collection of input features or attributes and target class pairs used to train network is called training set. The training set contains patterns not used for training purpose and is used to evaluate performance of network. Even with testing, model might over fit training data, causing degradation in performance when tested on new patterns. Cross validation is a useful technique which avoids this problem of over generalization by periodically testing model performance on cross validation data set during training phase. The best model is one with least cross validation error. This classification data set was split into training, cross validation and testing data sets with 60%, 20% and 20% memberships. Different network architectures were experimented with to determine best possible RFMLP-NN classifier. The most appropriate model chosen was two hidden layered RFMLP-NN with 12–12–10–6 architecture and hyperbolic tangent transfer functions at hidden layers as shown in figure 5. This network had best classification accuracy for testing data set. The training parameters for 12–12–10–6 RFMLP classifier is given in table 4.

4.5 Rough Fuzzy Multi Layer Perception Neural Network Job Scheduler

The Scheduler is developed based on output of RFMLP-NN. Given an input scheduling problem of size \(m \times n\), scheduler first maps each of \(mn\) operations to an input pattern according to classification scheme described in previous subsection. These \(mn\) input patterns are then presented to trained RFMLP-NN, which assigns priority index to each of them. The Giffler-Thomson algorithm which generates active schedules is used for schedule generation (Giffler et. al., 1960). An active schedule can be described as schedule in which no operation can be started earlier without delaying any other operation. The set of active schedules is guaranteed to contain an optimum
solution for any regular performance measure like makespan. The Giffler-Thomson algorithm iteratively chooses an operation to schedule from among set of available operations based on priority index. The operations are first ranked based on priority index assigned to each operation by RFMLP-NN. The operation with lowest rank is scheduled on identified machine. This process continues till all \( mn \) operations in problem have been scheduled. The Giffler-Thomson algorithm is an intelligent manufacturing, planning and scheduling paradigm and is implemented in C++ programming language. The software developed is generic in nature and could also generate schedules with other priority dispatching rules like shortest processing time etc.

Fig. 4. Block Diagram of the Procedure

Table 3. Sample Data for classification task

<table>
<thead>
<tr>
<th>Pattern – ID</th>
<th>Operation</th>
<th>Process Time</th>
<th>Remaining Time</th>
<th>Machine Load</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>First</td>
<td>Short</td>
<td>Long</td>
<td>Light</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Middle</td>
<td>Medium</td>
<td>Long</td>
<td>Light</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>61055</td>
<td>Later</td>
<td>Medium</td>
<td>Short</td>
<td>Heavy</td>
<td>4</td>
</tr>
<tr>
<td>61056</td>
<td>Last</td>
<td>Short</td>
<td>Short</td>
<td>Light</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 4. Training Parameters for 12-12-10-6 RFMLP Classifier

<table>
<thead>
<tr>
<th>Network Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step Size</td>
<td>0.01</td>
</tr>
<tr>
<td>Momentum Factor</td>
<td>0.7</td>
</tr>
<tr>
<td>Number of runs</td>
<td>10</td>
</tr>
<tr>
<td>Number of epochs per run</td>
<td>10000</td>
</tr>
<tr>
<td>Number of epochs without improvement in cross validation Error</td>
<td>500</td>
</tr>
</tbody>
</table>

Fig. 5. Rough Fuzzy Multi Layer Perception Neural Network (12–12–10–6)
5. RESULTS AND DISCUSSION

In this section, we discuss performance of RFMLP-NN classifier and scheduler on test set of problems.

5.1 Performance of Rough Fuzzy Multi Layer Perception Neural Network Classifier

Confusion matrices which shows desired classification (GA solutions) and output of classifier are compared on test data set given by table 5 are used to evaluate performance of RFMLP-NN classifier. In confusion matrix, diagonal entries represent number of test set instances classified correctly as set by GA for RFMLP-NN classifier. The classification accuracy for each class is calculated by dividing number of correct classifications by total number of instances. It can be observed from table that often deviation from correct classification is only by one class. This deviation can be attributed to presence of considerable noise in classification data set. The two main sources of noise are:

1) Genetic Algorithm Assignments: GA assigned different priorities to same operation in different schedules, as multiple sequences may produce same optimal makespan. This led to some ambiguity in classification data set with same training patterns having different target features.

2) Encoding of Classification Problem: The chromosome sequences are mapped according to classification scheme into training patterns. While this generalization reduced dimensionality of data which was desirable for comprehensibility, it represented loss of information and source of noise for classification task.

Table 5. Confusion Matrix of 12–12–10–6 RFMLP-NN Classifier

<table>
<thead>
<tr>
<th>Output/Desired Priority (Zero)</th>
<th>Priority (One)</th>
<th>Priority (Two)</th>
<th>Priority (Three)</th>
<th>Priority (Four)</th>
<th>Priority (Five)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority (Zero)</td>
<td>819</td>
<td>86</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Priority (One)</td>
<td>356</td>
<td>869</td>
<td>246</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>Priority (Two)</td>
<td>221</td>
<td>496</td>
<td>816</td>
<td>186</td>
<td>9</td>
</tr>
<tr>
<td>Priority (Three)</td>
<td>0</td>
<td>32</td>
<td>424</td>
<td>1119</td>
<td>532</td>
</tr>
<tr>
<td>Priority (Four)</td>
<td>0</td>
<td>0</td>
<td>66</td>
<td>319</td>
<td>119</td>
</tr>
<tr>
<td>Priority (Five)</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>56</td>
<td>521</td>
</tr>
<tr>
<td>Classification Accuracy (%)</td>
<td>64.79</td>
<td>66.29</td>
<td>59.46</td>
<td>86.56</td>
<td>28.46</td>
</tr>
<tr>
<td>Total Accuracy (%)</td>
<td>66.98</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.2 Schedule Generation and Comparison

A comparative summary of makespan of schedules generated by GA, RFMLP-NN, SPT and other priority dispatching rules for ft06 instance is given in table 6. The performance of Attribute Oriented Induction (AOI) rule set (Koonce et. al., 2000). AOI is rule induction method using concept hierarchies to aggregate membership of tuples that produced set of rules which assign priority to an operation based on either mean or mode placement. Makespan for schedules built by using dispatching rules other than SPT are from the empirical study (Käschel et. al., 1999). Only GA is able to achieve an optimal makespan of 55 units. RFMLP-NN developed in this work achieves makespan of 58 units, a deviation of 4 time units (5.45%) from optimum makespan. The deviations of other methods ranged from 12 to 29 units (21.8%–52.7%). The performance of RFMLP-NN is considerably better than performance of other methods in scheduling benchmark 6 × 6 problem. To access generalization capabilities of RFMLP-NN algorithm, test data set consisting of 10 randomly generated 6 × 6 problem scenarios is constructed. The motivation in using test set of 6 × 6 problem is to keep scheduling complexity similar to benchmark ft06 problem, while altering processing times and precedence constrains of operations. Schedules based on different approaches are built using Giffler-Thompson algorithm with priorities for operations being assigned from base algorithm (RFMLP-NN, AOI-Mean, AOI-Mode and SPT). Table 7 shows performance of these schedules on test data sets. GA solutions are considered benchmark solutions for comparing other schedulers (Yamada et. al., 1997). The figures indicated in bold in table 7 represent best solutions obtained for test instance under consideration. The RFMLP-NN scheduler provides better makespan (nearest to GA makespan) for more problem instance than other methods on test data set. Also, RFMLP-NN scheduler performed better than SPT heuristic in nine of ten cases. From an observation of average makespan values and percentage deviations provided in table 7, it is evident that RFMLP-NN scheduler approaches GA in scheduling test problems. This illustrates generalization capabilities of RFMLP-NN scheduler as learning part comes from only optimal solution to benchmark ft06 problem instance.

Analysis of Variance (ANOVA) technique is used for comparing performance of alternate schedulers. The experiment is designed as randomized complete block design to account for...
variability arising from different job scheduling problem instances in test data set. The assumptions made for this experiment are that observations are independent and normally distributed with same variance for each treatment (scheduler). The Anderson Darling test verified normality assumption (Lehmann, 1976). The Bartlett’s test is used to validate assumption of homogeneity of variances. The null hypothesis for this experiment, $H_0$ (considering that all treatment means are equal) is tested at 5% significance. If null hypothesis is rejected, we conclude that there exists a significant difference in treatment means.

Table 6. Makespan of Schedules for ft06

<table>
<thead>
<tr>
<th>Scheduler</th>
<th>Makespan</th>
<th>Deviation (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genetic Algorithm (GA)</td>
<td>55</td>
<td>0</td>
</tr>
<tr>
<td>Rough Fuzzy Multi Layer Perception-Neural Network (RFMLP-NN)</td>
<td>58</td>
<td>5.45</td>
</tr>
<tr>
<td>Attribute Oriented Induction (AOI)</td>
<td>67</td>
<td>21.81</td>
</tr>
<tr>
<td>Shortest Processing Time (SPT)</td>
<td>83</td>
<td>50.90</td>
</tr>
<tr>
<td>Most Work Remaining (MWKR)</td>
<td>67</td>
<td>21.81</td>
</tr>
<tr>
<td>Shortest Remaining Processing Time (SRMPT)</td>
<td>84</td>
<td>52.72</td>
</tr>
<tr>
<td>Smallest Ratio of Processing Time to Total Work (SPT-TWORK)</td>
<td>72</td>
<td>29.09</td>
</tr>
</tbody>
</table>

Table 7. Makespan obtained by various Schedulers on test data set
(The bold values indicate best solutions obtained for problem instance under consideration)

<table>
<thead>
<tr>
<th>Problem instance</th>
<th>GA</th>
<th>RFMLP-NN</th>
<th>AOI – Mode</th>
<th>AOI – Mean</th>
<th>SPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ft06-R1</td>
<td>46</td>
<td>48</td>
<td>49</td>
<td>53</td>
<td>54</td>
</tr>
<tr>
<td>ft06-R2</td>
<td>53</td>
<td>55</td>
<td>56</td>
<td>58</td>
<td>64</td>
</tr>
<tr>
<td>ft06-R3</td>
<td>60</td>
<td>60</td>
<td>62</td>
<td>67</td>
<td>71</td>
</tr>
<tr>
<td>ft06-R4</td>
<td>48</td>
<td>54</td>
<td>60</td>
<td>55</td>
<td>63</td>
</tr>
<tr>
<td>ft06-R5</td>
<td>55</td>
<td>60</td>
<td>61</td>
<td>63</td>
<td>66</td>
</tr>
<tr>
<td>ft06-R6</td>
<td>54</td>
<td>56</td>
<td>61</td>
<td>59</td>
<td>67</td>
</tr>
<tr>
<td>ft06-R7</td>
<td>51</td>
<td>54</td>
<td>53</td>
<td>53</td>
<td>60</td>
</tr>
<tr>
<td>ft06-R8</td>
<td>67</td>
<td>70</td>
<td>76</td>
<td>75</td>
<td>71</td>
</tr>
<tr>
<td>ft06-R9</td>
<td>54</td>
<td>56</td>
<td>59</td>
<td>68</td>
<td>59</td>
</tr>
<tr>
<td>ft06-R10</td>
<td>59</td>
<td>64</td>
<td>70</td>
<td>70</td>
<td>86</td>
</tr>
<tr>
<td>Average</td>
<td>54.7</td>
<td>57.7</td>
<td>60.7</td>
<td>62.1</td>
<td>66.1</td>
</tr>
<tr>
<td>Deviation (%)</td>
<td>-</td>
<td>5.48</td>
<td>10.97</td>
<td>13.53</td>
<td>20.84</td>
</tr>
</tbody>
</table>

The Duncan’s multiple range tests is then used to identify pairs of schedulers, which has significant difference in means. The treatment means are sorted in an ascending order. The test statistic is least significant studentized range $r_p$, where, $p$ denotes number of treatment means and depends on number of means and degrees of freedom. For $2 \leq p \leq 6$, values of $r_p$ are obtained from least significant studentized range table. The Duncan’s critical value, $R_p$ for these means is computed. The difference between pair of means drawn from set of ordered treatment means is compared with critical value, $R_p$. This comparison is carried out for all 15 possible combinations of treatment pairs. The determination of significant difference in means allowed Schedulers to be combined into groups as given in table 8.

The three groups identified by Duncan’s test correspond to different scheduling approaches for JSP. The optimization method viz. GA provided best makespan (Yamada et al., 1997). Machine Learning methods (RFMLP-NN, AOI-Mean and AOI-Mode) constituted second group B. While not statistically significant within this group, RFMLP-NN approach provided best results on test problems having lowest average makespan. Also, AOI based methods produce rules which assign priorities to operations. These rules are not sufficient to schedule any randomly generated $6 \times 6$ scenario. For some operations, input features of operation did not match rule antecedent and in such cases, an average priority index of 2.5 is assigned to such operations. In contrast to rule-based approach, trained RFMLP-NN could successfully describe any randomly generated $6 \times 6$ instance. The performance of all members in second group is better than SPT heuristic. RFMLP-NN scheduler has shown to have good generalization capabilities based on its performance on test set of 10 randomly generated $6 \times 6$ problem instance. Another dimension for evaluating performance of RFMLP-NN scheduler is its scalability on larger problem sizes. For this, five well-known problem instances from scheduling literature with sizes ranging from 100 to 400 operations were selected. These problems are ft10, la24, la36, abz7 and yn1 (Adams et al., 1988; Fisher et al., 1963; Lawrence, 1984; Yamada et al., 1992). The problems are selected from different sources to provide gradation.
in problem size. The makespan achieved by RFMLP-NN scheduler is compared to makespan of best known solution, GA and active schedulers based on SPT heuristic and random assignment heuristic as shown in Table 9. The last heuristic selects an operation from among set of schedulable operations randomly.

Analyses of results indicate a consistent pattern in performance of schedulers. GA provides best solution among different schedulers. RFMLP-NN scheduler outperforms other schedulers (SPT and Random) on all problems. The deviation in average makespan provided by RFMLP-NN scheduler is 11.60% from average makespan provided by GA, while those of SPT and random assignment heuristic are 41.56% and 65.43% respectively. The deviation of RFMLP-NN scheduler from average GA makespan increases by 6.12% points on larger problem data set from 5.48% on test set of 6 \times 6 problems. This compares favorably to increase in deviation of SPT scheduler by 20.42% points between two data sets. This is impressive because RFMLP-NN is trained on solutions obtained from single 6 \times 6 benchmark problem. To be able to provide good solutions with minimum computational effort on considerably larger problems with different sequencing constraints and processing times validates learning of RFMLP-NN. It shows that RFMLP-NN is able to capture certain key problem size invariant properties of good solutions as part of its training regimen. An inspection of result in Table 9 also reveals consistent divergence in GA performance when compared to best known solutions. This is because GA used in this research had simple evolutionary scheme with standard genetic operators and constants parameters. Typically, GA parameters have to be tuned with considerable experimentation. For larger JSP, GA has shown to achieve very good solutions with more sophisticated genetic operators and adaptive parameters (Yamada et al., 1997). In one such study, authors report that by incorporating local search into GA scheme, near optimal or optimal solutions are obtained on similar set of benchmark problems to those utilized in this work.

As the primary objective of this work is to develop and demonstrate potential of RFMLP-NN to learn from good solutions, we employed simple GA which provided optimal solutions for 6 \times 6 problem. The source of solutions for learning task is not important and is indeed one of the strengths of this approach. In real world scenario, solutions considered as good solutions by experts can be used to train RFMLP-NN thus providing computational model based on expert knowledge. Alternately, more effective optimizers could be used to generate good solutions for training RFMLP-NN. RFMLP-NN scheduler developed in this work is generic in nature and could be effectively combined with any optimizer to generate good solutions with low computational effort and time. Though RFMLP-NN scheduler is computationally less intensive than GA and offers more comprehensible scheduling approach. It also provides an attractive alternative to simple heuristics like SPT as it effectively utilizes more problem specific knowledge in making scheduling decisions with similar computational effort. For an arbitrary problem, development of an efficient optimizer involves significant design effort and an understanding of problem domain. The learning frameworks using RFMLP-NN presented in this work are generic and can be easily applied whenever there is known good solutions to problems, irrespective of their source. A thorough understanding of problem domain is not essential for successful application of methodology. Thus, learning framework is also particularly suited to problem domains in which current understanding is limited.

6. CONCLUSION
This paper presents a novel knowledge based approach viz., RFMLP-NN to solve JSP by utilizing various ingredients of Machine Learning paradigm. The ability of GA to provide multiple optimal solutions is exploited to generate knowledge base of good solutions. RFMLP-NN is successfully trained on this knowledge base. RFMLP-NN scheduler is utilized to schedule any job scenarios of comparable size to the benchmark problem. Also, this methodology can be employed for learning from any set of schedules, regardless of their origin. A test problem set consisting of 10 randomly generated $6 \times 6$ scenarios is used to evaluate generalization capabilities of the developed Scheduler. A comparative evaluation of RFMLP-NN scheduler with other schedulers developed from a different Machine Learning methodology and SPT heuristic is also undertaken. Among these schedulers, RFMLP-NN scheduler developed in this work has closest average makespan to that of GA. RFMLP-NN scheduler also performed satisfactorily on test set of larger problem sizes, which demonstrates success of RFMLP-NN in learning key properties to identify good solutions. Thus, in this investigation a RFMLP-NN scheduler is developed which provides an appreciable performance for JSP.

REFERENCES


Fisher, H. and Thompson, G. L. (1963). Probabilistic learning key properties to identify good solutions. Thus, in this investigation a RFMLP-NN scheduler is developed which provides an appreciable performance for JSP.

REFERENCES


Fisher, H. and Thompson, G. L. (1963). Probabilistic learning key properties to identify good solutions. Thus, in this investigation a RFMLP-NN scheduler is developed which provides an appreciable performance for JSP.

REFERENCES


Luangpipoon, P., and Boonvirojrit, A. (2004). Comparison of heuristic methods for the n-job and m-machine flowshop with resource constraints, Department of Industrial Engineering, Faculty of Engineering, Thammasat University, Pathumthani, Thailand.


AUTHORS PROFILE

Arindam Chaudhuri is Lecturer in Computer Science Engineering at Birla Institute of Technology Mesra, Patna Campus, India. He holds Bachelors degree in Physics from Calcutta University, Master degree in Information Systems and Operations Research from All India Management Association, New Delhi, Master degree in Computer Applications from Networked University, Chattisgarh and Master of Technology in Computer Science Engineering from Doeacc Society, New Delhi. He is working towards PhD degree in Computer Science from Netaji Subhas Open University, Kolkata, India. His research interests include Soft Computing and Optimization Algorithms.

Kajal De is Professor in Mathematics at Netaji Subhas Open University, Kolkata, India. She holds Bachelors, Masters and PhD degrees in Applied Mathematics from Calcutta University. She has over 15 years of teaching and research experience and has published papers in leading International Journals in Applied Mathematics. Her research interests include Fuzzy Sets and Optimization Algorithms.