**Neural network architecture**

Each job and each batch correspond to a computational node in the neural network. The job nodes are presented by J1, J2, … , Jn and construct the job layer, which corresponds to the input layer of the neural network. The batch nodes are presented by B1, B2, … , Bn and construct the batch layer, which corresponds to the hidden layer of the neural network. We also consider an initial node and a

final node. The initial node counts the number of iterations and epochs. The final node placed

in the output layer to capture the output of the job and the batch nodes. In the HNN, the nodes in different layers are connected by some arcs.

In the job layer, nodes receive an input signal from the initial node. The job nodes are also fully connected to the nodes of the batch layer through arcs. The arcs are characterized by weights w1, w2, … , wn. The jobs are assigned to the batches based on these weights. In each iteration,

jobs compete for assignment; one job wins and is assigned to one batch. All nodes in the job

layer are also linked to the final node. In the batch layer, each batch node is connected to the

next batch node to send a signal when it cannot fit a given job. When a batch accepts a job, it

sends a signal to the job node and also sends the index of assigned job to the final node. The

input, activation, and output functions are defined for all the nodes. Fig 1 shows the architecture of the proposed neural network.

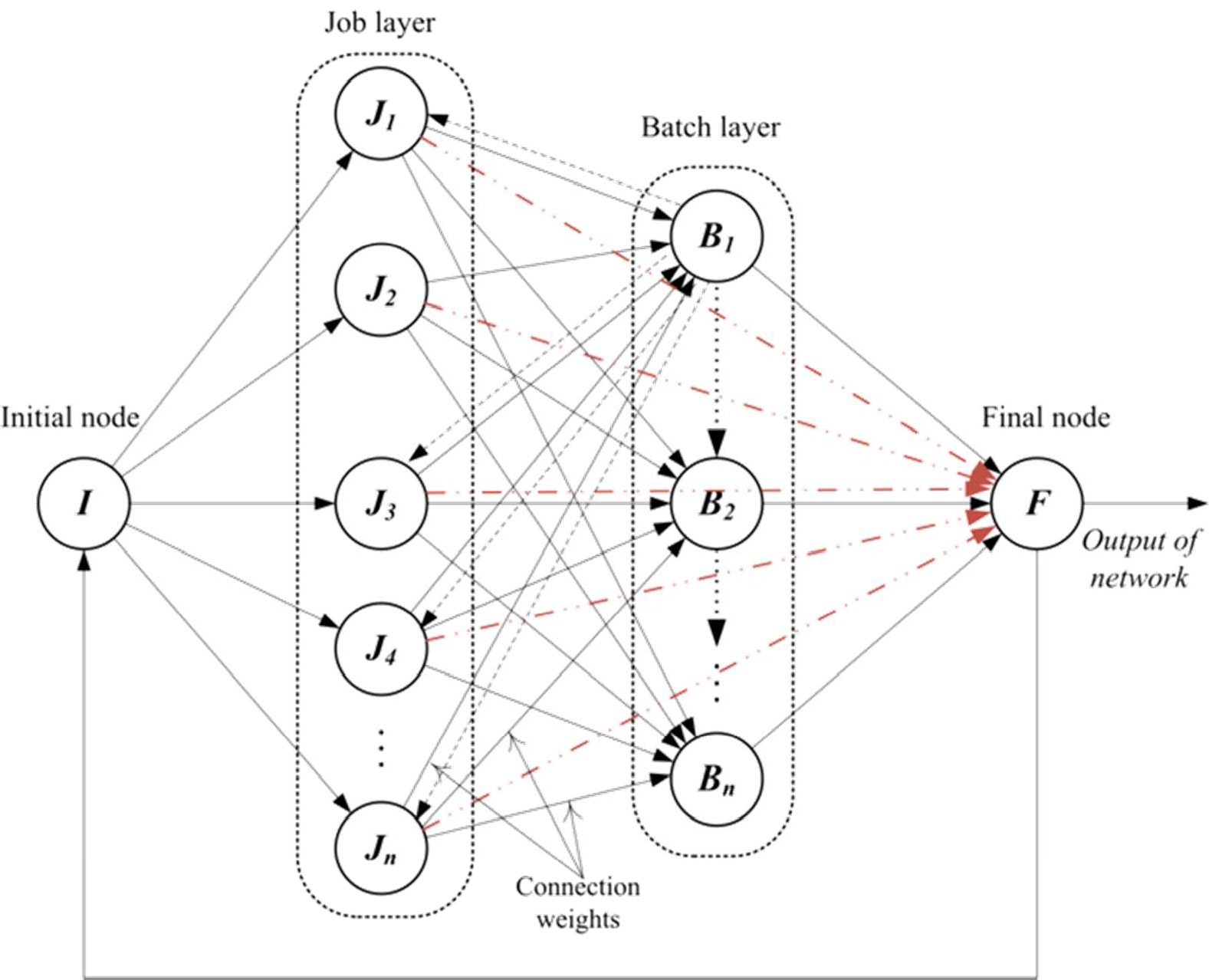


Fig. 1. Architecture of the proposed neural network

**Notations**

k: Epoch number, k = 1, 2, … , kmax

t: Iteration number, t = 0, 1, 2, … , n

J: Set of jobs, 1, … , n

B: Set of batches, 1, … , n

I: Initial node

F: Final node

Jj: jth job node in the job layer, j ∈ J

UJ: Set of unassigned jobs

Bb : bth batch node in the batch layer, b ∈ B

LB: Lower bound of the problem

α: Learning rate

pt(j): Processing time of Job j

s(j): size of Job j

rt(j): Release time of Job j

dd(j): Due-date of job j

All the functions required for the Initial, Job, Batch, and Final nodes in iteration t are as

follows.

***Initial node:***

IFI(t):Input function

AI(t):Activation function

OFI(t):Output function

***Job nodes:***

IFJj(t):Input function of jth job node, j ∈ J

AJj(t):Activation function of jth job node, j ∈ J

OFJB (t):Output function of jth job node to the batch nodes, j ∈ J

OFJFj(t):Output function of jth job node to the final node, j ∈ J

***Batch nodes:***

IFBjb(t):Input function from jth job node to bth batch node, j ∈ J, b ∈ B

AB(t): Activation function of bth batch node, b ∈ B

OFBJbj(t):Output function of bth batch node to jth job node, j ∈ J, b ∈ B

OFBBb(t):Output function of bth batch node to (b + 1)th batch node, b ∈ B, b ≠ n

OFBFb(t):Output function of bth batch node to the final node, b ∈ B

***Final node:***

IFFJ(t):Input function of the final node from the job nodes

IFFBb(t):Input function of the final node from the bth batch node, b ∈ B

AF(t):Activation function of the final node

OFFI(t):Output function of the final node to the initial node

Functions which are calculated at the end of epoch *k* are defined as follows:

OFF(k):Output function of the final node (the obtained solution in epoch *k*)

wj(k):Weight on the arcs from jth job node to the batch nodes, j ∈ J

e(k):Error or gap between the solution and the lower bound in epoch *k*

**Description of the functions**

The lower bound for Total weighted tardiness is set to 0.The initial weights for all nodes are assigned to 1 .

*Initial node:*

At the beginning of the algorithm, set t = 0 and k = 1. Define the input function of the

initial node as follows:

1 t = 0

IFI t = {

OFFI(t) t > 0

At the beginning of the first iteration of the first epoch, the input function of the initial

node is set to 1. Thereafter, its value is determined by the output function of the final node.

The activation function of the initial node is defined as follows:

t = t + 1, k = k if IFI(t) = 1

AI(t) = {t = 1, k = k + 1 if IFI(t) = 2

t = 0, k = 1 if IFI(t) = 3

The activation function of the initial node counts the number of iterations and epochs. IFI

of 1 implies that all the jobs have not been assigned and the network should run a new

assignment iteration at the current epoch. IFI of 2 indicates that all the jobs have been

assigned and the network should run a new epoch. When IFI is 3, the network should stop.

After computing the activation function, the output function of the initial node is calculated

by the following equation:

OFI(t) = {0 t = 0

1. t > 0

OFI of 1 indicates that a new assignment iteration should be executed. In other words, the

problem still needs to be solved until all the jobs are assigned to the batches.

*4.3.2. Job layer*

The input function for each job nodes in the job layer is defined as the output function of the initial node based on the following equation:

IFJ (t) = OFI(t) ∀j ∈ J.

j

The activation function of the job nodes in the job layer is defined as follows:

For t = 0 and ∀j ∈ J: AJj(0) = 1.

For t > 0 and ∀j ∈ J:

0 if AJj(t − 1) = 0 or (AJj(t − 1) = 1 and OFBJbj(t) = 1)

AJj(t) = {

1. if AJj(t − 1) = 1 or (AJj(t − 1) = 0 and IFI(t) = 2)

For each job j, AJj indicates the job j should be assigned or not. AJj of 0 shows that job j has been assigned in pervious iterations and there is no need to be assigned in the next iterations of the current epoch. AJj of 1 implies that job j has not been assigned yet and should be assigned in the next iterations.

The output function of the job nodes to the batch nodes for all j ∈ J, is defined as follows:

OFJBj(t) = Factor(f)∗ wj(k) ∗ AJj(t)

Here the 8 different factors are considered as follows:

1.LPT=pt(j)

2.Size=s(j)

3.ERT=1/(rt( j))

4.EDD=1/(dd(j))

5.FDD=1/(rt(j)+pt(j))

6.ODD=1/(rt(j)+3\*pt(j))

7.LST =1/(dd(j)-pt(j))

8.CI=s(j)/(dd(j)-pt(j))

If job j is not assigned yet, i.e. AJj(t) = 1, OFJBj sends a weighted signal to each batch

node, otherwise it is set to 0. Note that the output function of a job node to all the batch nodes

is the same.

The output function of the job nodes to the final node for all j ∈ J, is defined as follows:

1 if AJj(t) = 0

OFJB(t)={

0 if AJj(t) = 1

The output function of job j to final node indicates that job j has been assigned or not.

*4.3.3. Batch layer*

In this section, input, activation and output functions of the batch nodes are explained.The job with the maximum of OFJBj(t) is selected and placed in the first batch with enough space to accommodate it. If it fits in no existing batch, a new batch is constructed and the job is placed into the new batch. Define RCb(t) as the residual capacity of bth batch in iteration t. In the first iteration of each epoch, the residual capacity of each batch is equal to the machine capacity, i.e.RC (1) = C and the activation function of the first batch is set to 1, AB (1) = 1. For the other batches, the activation function is initialized to 0, ABb(1) = 0, ∀ b ∈ B, b ≠ 1. It indicates that only the first batch is open at the beginning of each epoch. The activation function for t > 1 is defined as follows:

0 if ABb(t − 1) = 0 or IFI(t) = 2

ABb(t) = {1 if ABb(t − 1) =1or (ABb(t) =0and OFBBb–1(t) =1)

1. if ABb(t − 1) = 1 and RCb(t) < Min{sS|l ∈ UJ}

The value of 0 for ABb shows that batch b has not been opened yet. When batch (b − 1) cannot fit the job with maximum OFJBj, it sends a signal to batch b, so batch b opens, signified by ABb of 1. When an open batch’s residual capacity is less than the minimum size of unassigned jobs, then the batch closes (ABb = 2).

The input function for all the batch nodes is defined as follows:

Max(OFJBj(t)) if ABb(t) = 1

IFB(t)={

1. if ABb(t) = 0 or ABb(t) = 2

The input function of each batch is equal to the maximum output of the job nodes, when the batch is open. Otherwise, if the batch is close or not yet open, it cannot accept any job.Assume ℎ is the index of job with Maxj(OFJBj(t)). Then job ℎ is assigned to batch b if its size is less than the residual capacity of batch b. Assignhb(t) represents the assignment of job ℎ to batch b and is defined as follows.

1 if sh ≤ RCb(t)

Assign(t)={

1. Otℎerwise

After assigning job ℎ to batch b, the residual capacity of batch b decreases by sh, i.e. RCb(t) = RCb(t) − sh.For each node in the batch layer, we define three output functions:

- From the batch node to the job node: Batch b sends a signal of 1 to job ℎ, if it accepts job ℎ, otherwise it sends 0.

1 if Assignhb(t) = 1

OFBJbh(t) = {

1. otℎerwise

- From the batch node to the next batch node: Batch b sends a signal to batch (b + 1), when it has not enough capacity to accept job ℎ, where ℎ is the index of job with Maxj(OFJBj(t)).

1 if ABb(t − 1) = 1 and RCb(t) < sh

OFBBb(t) = {

1. Otℎerwise

- From the batch node to the final node: When a job is assigned to batch b, the batch sends the index of assigned job to the final node. OFBFb of 0 implies that no job has been assigned to batch b in the current iteration.

ℎ if Assignhb(t) = 1

OFBFb(t) = {

0 Otℎerwise

***Final node:***

In the final node, two input functions are defined:

- To the final node from the job nodes: The input function of the final node from the job

Nodes (*IFFJ*) calculates the number of assigned jobs so far, i.e.,

IFFJ(t) = ∑nOFJFj(t). If all the jobs are assigned, *IFFJ* is equal to the number of jobs

- To the final node from the batch nodes: The input function of the final node from the

bth batch node (IFFBb) returns the index of assigned job to the bth batch in the current

iteration, i.e., IFFBb(t) = OFBFb(t) .

The Activation function of the final node is defined as follows:

0 if IFFJ(t) < n

AF(t) = {

1 if IFFJ(t) = n

AF of 1 indicates that all the jobs are assigned and AF of 0 implies that all the jobs are not

assigned till now.

For computing the objective function, we should define an assignment matrix, which represents the assignment of jobs to the batches. At the beginning of each epoch, the Assignment Matrix (AM) is initialized to an n × n zero-matrix. After each assignment iteration, AM is modified as follows, where b is the index of batch in which IFFBb ≠ 0.

AM(IFFBb(t), b) = 1

AM is a binary matrix which AM(j, b) = 1 indicates that the job j has been assigned to the batch b. So the bth column of AM represents the assigned jobs to the bth batch. At the end of each epoch when all the jobs have been assigned to the batches, i.e. AF(t) = 1, the output of the network should be calculated. The output function OFF(k) represents the total weighted tardiness of the jobs.

The final node sends a signal to the initial node:

1 if AF(t) = 0

OFFI(t) = {2 if (AF(t) = 1) and (k < kmax)

3 if [(AF(t) = 1) and (k = kmax)] or [(AF(t) = 1) and (OFF(k) = LB)]

If all the jobs have not been assigned yet, a new assignment iteration should be executed in the current epoch, signified by OFFI of 1. OFFI of 2 indicates that all the jobs have been assigned but the maximum number of epochs (knas) is not reached, so the network should run another epoch. If either the maximum number of epochs is reached or a lower bound solution is found, the stopping rule is satisfied, signified by OFFI of 3. At the end of each epoch, by finding a new solution OFF(k), the error term is calculated as the gap between the obtained solution and the lower bound. The best solution and the best corresponding weights are updated if needed. If the maximum number of epochs is not reached, the weights are also modified through the learning strategy and the network will run a new epoch.

49

***Learning strategy***

As mentioned earlier, the hybrid neural network approach employs a priority rule heuristic, and the improvements occur iteratively through a learning strategy involving weights modification. Therefore the search problem in the solution space is transformed into a problem of search for the best weight vector in the weight space. Weights are modified once at the end of each epoch, they are not modified from one assignment iteration to the next. The weights vector provides a mechanism for perturbation in the order of jobs which leads to a new solution. If the error term is too high in an epoch, then the order in which the jobs are assigned to the batches should be changed more comparing to the low error case. The intention of weights modification is to consider this concept. If the error is low, the new solution is obtained in the local neighborhood of the old solution. When the solution is relatively close to the optimal, the error term decreases and therefore the magnitude of change in the weights also decreases and the convergence happens. Using FFLPT as the heuristic guarantees that always a feasible solution is found for each order of jobs. In the proposed HNN approach, the weights are modified by the search strategy proposed by Agarwal (2009) as follows:

wj(k) + (α ∗ e(k) ∗ R) if R < 0.5

wj(k + 1) = {

wj(k) − (α ∗ e(k) ∗ R) if R > 0.5

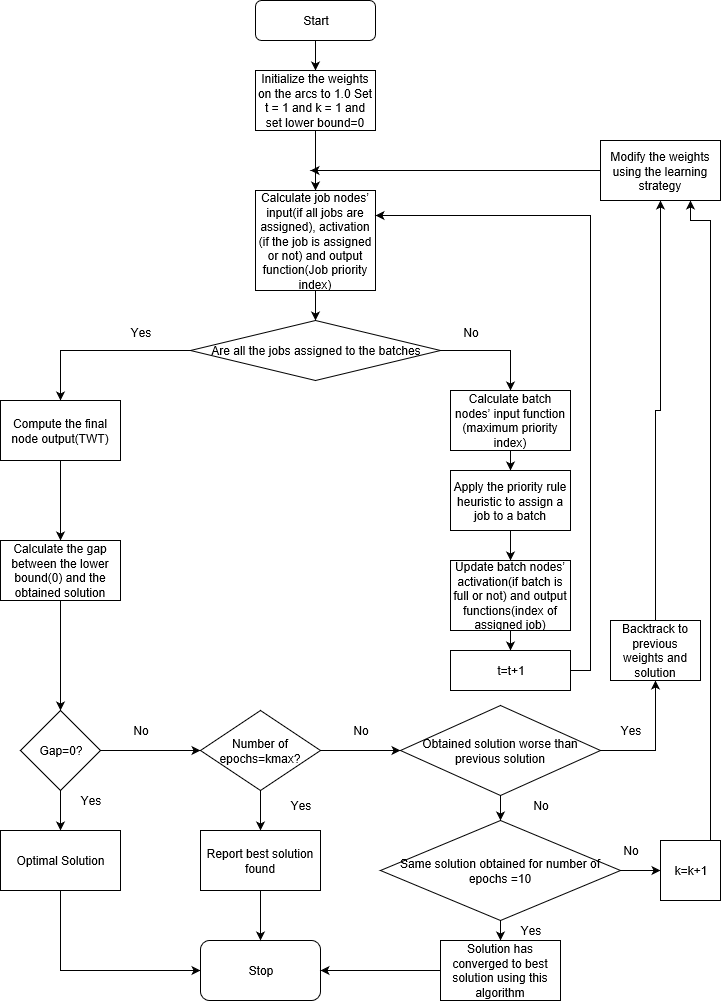
where R is a uniform random number between 0 and 1.

In addition, reinforcement and backtracking mechanisms are applied to improve the solution quality. When the HNN approach finds a better solution compared to the previous epoch, the weights are reinforced. This reinforcement mechanism helps to preserve the relative weights of the jobs for a few epochs. The weights are magnified by the following strategy:

wj(k) = wj(k) + F ∗ (wj(k) − wj(k − 1)) ∀ j ∈ J

where F is the reinforcement factor and it can be any real positive number. In this research, based on the computational experiments the reinforcement factor is set to 4.

When after a certain number of epochs, the solution does not change; the network backtracks to the previous best solution and best weights. Such backtracking prevents the network from continuing a path of no improvement anymore. In this research, when the same solution is obtained more than 10 times then it can be inferred that the HNN cannot improve any further. Therefore this is the stopping criteria.

Flowchart of the proposed algorithm