Validation of the Improved Incremental Assignment Algorithm

Pinzhi Wang*, Youssou Gningue*
Laurentian University, Ontario, CANADA

Haibin Zhu†
Nipissing University, Ontario, CANADA

Abstract

The Assignment Problem is a basic combinatorial optimization problem. In a weighted bipartite graph, the Assignment Problem is to find the largest sum of weights matching. The Hungarian method is a well-known algorithm, which is combinatorial optimization. Adding a new row and a new column to a weighted bipartite graph is called the Incremental Assignment Problem (IAP). The algorithm of the Incremental Assignment Problem utilizes the given optimal solution (the maximum weighted matching) and the dual variables to solve the matrix after extending the bipartite graph. This paper proposes an improvement of the Incremental Assignment Algorithm (IAA), named the Improved Incremental Assignment Algorithm (IIAA). The improved algorithm will save the operation time and operation space to find the optimal solution (the maximum weighted matching) of the bipartite graph.

Key Words: Assignment problem, weighted bipartite graph, Hungarian algorithm, incremental assignment problem.

1 Introduction

A matching or independent edge set in a graph is a set of edges without common vertices. There are several algorithms of the matching problem, such as matching in weighted bipartite graphs, matching in unweighted graphs, matching in general graphs. In unweighted graphs, weighted bipartite graphs and matching in unweighted graphs, maximum cardinality matching is sought [7]. In matching in weighted bipartite graphs, each edge has an associated value. A maximum weighted bipartite matching is defined as a matching where the sum of the values of the edges in the matching has a maximal value. If the graph is not complete bipartite, missing edges are inserted with value zero. Finding such a matching is known as the assignment problem. A common variant consists of finding a minimum-weighted perfect matching [2]. In this research, we are interested in the maximum weighted matching problem in bipartite graphs. The well-known algorithm for the assignment problem is the Kuhn-Munkres algorithm or the Hungarian algorithm, originally proposed by Kuhn in 1955 [3] and refined by Munkres in 1957 [4]. This algorithm has $O(n^3)$ complexity when it is implemented with proper data structures.

The incremental assignment problem is described as given a weighted bipartite graph and its maximum weighted matching, determine the maximum weighted matching of the graph extended with a new pair of vertices, one on each partition, and weighted edges connecting these new vertices to all the vertices on their opposite partitions [5]. It can be solved with the Hungarian Algorithm as the ordinary assignment problem. But in [6], Toroslu and Üçoluk propose an algorithm that utilizes the given maximum weighted matching of the maximum-weighted-matched part of the bipartite to determine the maximum weighted matching of the whole (extended) bipartite graph. The complexity of the algorithm is $O(n^2)$.

Consider the situation that there will be thousands or millions of weights. It is costly to calculate the extended feasible labels by using the incremental assignment problem algorithm. The goal of this paper is to present an algorithm to improve the incremental assignment problem algorithm, which reduces the complexity to $O(n)$ in four situations.

2 Background

A graph $G = (V, E)$ is bipartite if there exist two disjoint partitions X and Y $(V = X \cup Y, X \cap Y = \emptyset)$ and no edge connects vertices in the same partition $(E \subseteq X \times Y)$. A matching M is a subset of the edges E $(M \subseteq E)$, such that $\forall v \in V$ at most one edge in M is incident upon v. The size of matching is $|M|$, the number of edges in M. A maximum matching is a matching of maximum size (maximum number of edges). In a maximum matching, if any edge is added to it, it is no longer matching [1].

A weighted bipartite graph $G = (X \cup Y, X \times Y)$ is the graph in which rows correspond to the X partition and columns correspond to the Y partition of vertices. Each entry W_{ij} represents the weight of the edge between the vertices X_i and Y_j. The weight of matching M is the sum of the weights of edges in M.

Given a matching M, an alternating path is a path that begins with an unmatched vertex and whose edges belong alternately to the matching and not to the matching. An augmenting path is an alternating path that starts from and ends on free (unmatched) vertices. All alternating paths originating from a given unmatched node form a Hungarian tree.

*pwang@laurentian.ca, ygningue@laurentian.ca.
†haibinz@nipissingu.ca.

In Figure 2, let M be a matching of G.Vertex v is matched if it is an endpoint of edge in M; otherwise, v is free. $Y_2, Y_3, Y_4, Y_6, X_2, X_4, X_5, X_6$ are matched, other vertices are free. Y_5, X_6, Y_6 is an alternating path. $Y_1, X_2, Y_2, X_4, Y_4, X_5, Y_3, X_3$ is an augmenting path.

3 Improved Incremental Assignment Algorithm

The original matrix adds a new pair of vertices to the maximum-weighted-matched bipartite graph, in which feasible vertices and the maximum weighted matching are given. Any feasible label to the newly added pair of vertices is assigned. By using these labels, we can determine the maximum weighted matching of the whole extended bipartite graph. Before presenting the algorithm, we study the different cases where a solution can be derived in a very fast manner.

3.1 Analysis of the Different Cases

With the adding of vector row and vector column $n+1$ and their respective costs, we can evaluate the marginal change associated with each row and column. The final values of the dual variables α_i and β_j for $i \in 1 \dots n$ and $j \in i \dots n$. This provides the evaluation of the differences for each row DR_j and column DC_i

$$DC_i = W_{i,n+1} - \alpha_i \quad and \quad DR_j = W_{j,n+1} - \beta_j$$

This allows us to evaluate maximal values of each column DCK and row DRT

$$DCK = max_{1 \le i \le n}(W_{i(n+1)} - \alpha_i) = DC_k \quad and \quad DRT = max_{1 \le j \le n}(W_{(n+1)j} - \beta_j) = DR_t$$

Their respective number (LC and LR) are also evaluated by

$$LC = Card(Argmax_{1 \le i \le n}(W_{i(n+1)} - \alpha_i)) \quad and \quad LR = Card(Argmax_{1 \le j \le n}(W_{(n+1),j} - \beta_j))$$

CASE I. When the following relation is satisfied

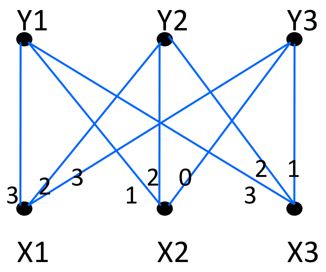
$$W_{n+1,n+1} \ \ge \ DCK + DRT$$

They are presented below.

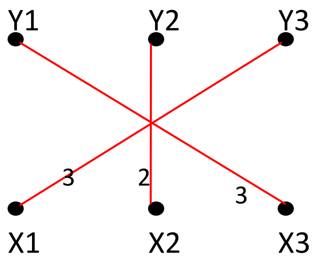
CASE II. When there is only one-row k and column t reaching the maximal marginal changes with $X_{k,t} = 1$ i.e.

$$LC = LR = 1 \text{ and } X_{k,t} = 1$$

This means that the crossing variable of row k and column t is assigned in the optimal solution of the n assignment problem. Setting $X_{k,t} = 0$ the $n+1$ assignment problem frees $X_{k,n+1}$ and $X_{n+1,t}$. Then by setting $X_{k,n+1} = 1$, and $X_{n+1,t} = 1$,

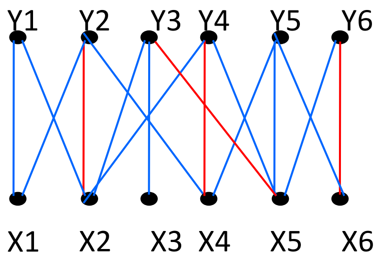


bipartite graph

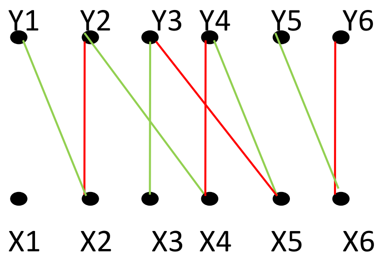


bipartite weighted graph

Figure 1: Bipartite graph and bipartite weighted graph



bipartite graph



alternating/ augmenting path

Figure 2: Bipartite graph and alternating/ augmenting graph

the maximal possible increase of the objective function is reached. This provides an optimal solution to the $n+1$ assignment problem.

CASE III. When there is only one-row k and column t reaching the maximal marginal changes with $X_{k,t}=0$ i.e.

$$LC = LR = 1 \text{ and } X_{k,t} = 0$$

Moreover, we have a column s and a row r such that

$$X_{k,s} = 1 \text{ and } X_{r,t} = 1 \text{ with } C_{rs} = \alpha_r + \beta_s$$

This means that the crossing variable of row k and column t is not assigned in the optimal solution of the n assignment problem. However, the complementary column s of the basic variable on row k and the complementary row of the basic variable on column t yield a variable X_{rs} which satisfies $C_{rs} = \alpha_r + \beta_s$. Since X_{rs} in the Equality Graph, setting $X_{k,s} = 0$ *and* $X_{r,t} = 0$. The $n+1$ assignment problem frees $X_{k,n+1}$ and $X_{n+1,t}$. Then by setting $X_{k,n+1} = 1$, and $X_{n+1,t} = 1$, the maximal possible increase of the objective function is reached. This provides an optimal solution to the $n+1$ assignment problem

CASE IV. When there is more than one row or more than one column reaching the maximal marginal changes

$$\text{if } max(LC, LR) > 1$$

This case presents two subcases

$$max(LC, LR) = LR \geq LC \text{ or } max(LC, LR) = LC > LR$$

SUBCASE IV-a. The subcase is encountered when

$$max(LC, LR) = LR \geq LC \text{ and } LR > 1$$

For each $j = 1, \dots, LR,$ we find the basis variable $X_{r\sigma(j)} = 1$ on the column $\sigma(j)$.

Then we test if $DC_k = (W_{r(n+1)} - \alpha_r) = DC_r$. This means that r and $\sigma(j)$, satisfies the second condition of Case II. We obtain an optimal solution by setting

$$X_{r\sigma(j)} = 0,\ X_{(n+1)\sigma(j)} = 1 \text{ and } X_{(n+1)\sigma(j)} = 1$$

SUBCASE IV-b. The subcase is encountered when

$$max(LC, LR) = LC > LR \text{ and } LC > 1$$

For each $i = 1, \dots, LR,$ we find the basis variable $X_{\theta(i)s} = 1$ on the column $\theta(i)$.

Then we test if $DR_t = (W_{(n+1)s} - \beta_s) = DR_s$. This means that $\theta(i)$ and s satisfy the second condition of Case II. We obtain an optimal solution by setting

$$X_{\theta(i)s} = 0\ X_{\theta(i)(n+1)} = 1 \text{ and } X_{(n+1)s} = 1$$

3.2 Algorithm Presentation

This algorithm is adopted from the Hungarian algorithm, which uses the feasible labels of the vertices together with the maximum weighted matching.

Improved Incremental Assignment Algorithm:
Input:

- n assignment problem comprising a bipartite graph, $\{V, E\}$ (where $V = X \cup Y, X \cap Y = \emptyset,\ |X| = |Y| = n+1$) and $(n+1) \times (n+1)$ matrix of edge weights W_{ij}
- An optimal solution to the $n \times n$ sub-problem of the above assignment problem, comprising a matching M^* of the first n nodes of X to the first n nodes of Y, and the final values of the dual variables α_i and β_j for $i \in 1 \dots n$ and $j \in i \dots n$

Output: An optimal matching M, for the $(n+1) \times (n+1)$ problem.

```
1. Perform initialization:
    (a) Find the difference of each role and each column as follows:
                    $DCK = -10^6$
    For $i = 1, \dots, n$ do
    S et $t = \sigma(1)$        $DC_i = \mathrm{W_{i,n+1}} - \alpha_\mathrm{i}$
        If $DCK < DC_i$ then set $DCK = DC_i$ and $LC=1$; $\theta(1) = i$
        Else If $DCK = DC_i$ then $LC = LC + 1$; $\theta(LC) = i$

            Else ($DSK$ did not change)
            EndIf

        EndIf

    EndFor

    Set $k = \theta(1)$
    The previous procedure provides
```

$$DCK = \max_{1 \le i \le n}(W_{i(n+1)} - \alpha_i) = DC_k$$

$$LC = Card(Argmax_{1 \le i \le n}(W_{i(n+1)} - \alpha_i))$$

$$DRT = -10^6$$

```
    For $j = 1, \ldots, n$ do
```

$$DR_j = W_{j,n+1} - \beta_j$$

```
        If $DRT < DR_j$ then set $DRT = DR_j$ and LR=1; $\sigma(1) = j$
        Else If $DRT = DR_j$ then $LR = LR + 1$; $\sigma(LR) = j$

            Else ($DRT$ did not change)
            EndIf

        EndIf

    EndIf

    Set $t = \sigma(1)$
    The previous procedure provides
```

$$DRT = max_{1 \le j \le n}(W_{(n+1)j} - \beta_j) = DR_t$$

$$LR = Card(Argmax_{1 \le j \le n}(W_{(n+1),j} - \beta_j))$$

$$I = DCK + DRT$$

```
    (b) If $I \le W_{(n+1)(n+1)}$, then

            $X_{(n+1)(n+1)} = 1$, then go to step 3

        Else If $LC = LR = 1$ then

            If $X_{kt} = 1$,
            Set $X_{k(n+1)} = 1$ and $X_{(n+1)t} = 1$,

                Then go to step 3.

            Else ($X_{kt} \neq 1$)

                Find the complementary column $s$ of the basic variable on row $k$
                Find the complementary row $r$ of the basic variable on column $t$
                If $X_{rs}$ satisfies $C_{rs} = \alpha_r + \beta_s$ (Equality Graph) then
                Set $X_{rs} = 1$ then $X_{k(n+1)} = 1$ and $X_{(n+1)t} = 1$
                Then go to step 3

            Else go to step (c).

    Else if $max(LC, LR) > 1$

        If $max(LC, LR) = LC$   then

            For $j = 1, \dots, LR$ do

                Find the basis variable $X_{r\sigma(j)} = 1$
                If $DC_k = (W_{r(n+1)} - \alpha_r)$ then set

                  $X_{r\sigma(j)} = 0$, $X_{(n+1)\sigma(j)} = 1$ and $X_{r(n+1)} = 1$

                  Go to step 3

            End if

        EndFor

    EndIf

  Else if $max(LC, LR) > 1$

    If $max(LC, LR) = LR$   then

        For $i = 1, \dots, LC$ do
        Find the basis variable $X_{\theta(i)s} = 1$
        If $DR_t = (W_{(n+1)s} - \beta_s)$ then set

              $X_{\theta(i)s} = 0$, $X_{\theta(i)(n+1)} = 1$ and $X_{(n+1)s} = 1$

            Go to step 3

        End if

    EndFor

  EndIf

EndIf
```

(c) Incremental assignment algorithm: Assign feasible values to the dual variables α_{n+1} and β_{n+1} as follows:

$$\beta_{n+1} = max\left(DC_k, W_{(n+1)(n+1)}\right)$$

$$\alpha_{n+1} = DR_t$$

2. Perform the Stage from the basic Hungarian algorithm detailed in the Hungarian Algorithm.

3. Output the resulting matching M.

4 Implementation and Performance Experiments

When properly implemented, the Incremental Assignment Algorithm (IAA) can operate with the computational complexity of $O(n^2)$ [5]. To verify the performance of the Improved Incremental Assignment Algorithm (IIAA), a program is implemented based on that of the IAA.

Five cases are designed for the Improved Incremental Assignment Algorithm (IIAA) in 100 dimensions (a matrix with 100×100 elements) and 200 dimensions (a matrix with 200×200 elements). Case I shows the operation time of the circumstances that $X_{kt} = 1$ and $I \leq W_{(n+1)(n+1)}$. Case II shows the running time of the matrix that $X_{kt} = 1$ and $I > W_{(n+1)(n+1)}$. Case III displays the operation time when $X_{kt} \neq 1$. Find the complementary column s of the basic variable on row k and the complementary row of the basic variable on column t, if X_{rs} satisfies $C_{rs} = \alpha_r + \beta_s$. Case IV reveals the running time when we have more than one largest difference of the row and more than one largest difference of the column. Case V is a general case, which expresses that the random matrix can be solved by the IIAA.

All the experiments are workable in both algorithms. Matrix is formed by randomly creating weights. The test takes 100 random new pairs of vertices in each dimension matrix.

$$\text{Chances} = \frac{\text{Numbers of each case}}{\text{Total number of the cases}} \times 100$$

Figure 3 provides the trend line of chances that a random matrix is solved by the Improved Incremental Assignment Algorithm (IIAA).

Figure 4 presents the average time of different cases, which have been solved by the improved incremental assignment algorithm and incremental assignment algorithm. The time unit is millisecond.

Table 1 presents the average operation time of different pairs of vertices, which have been solved by improved incremental assignment algorithm and incremental assignment algorithm

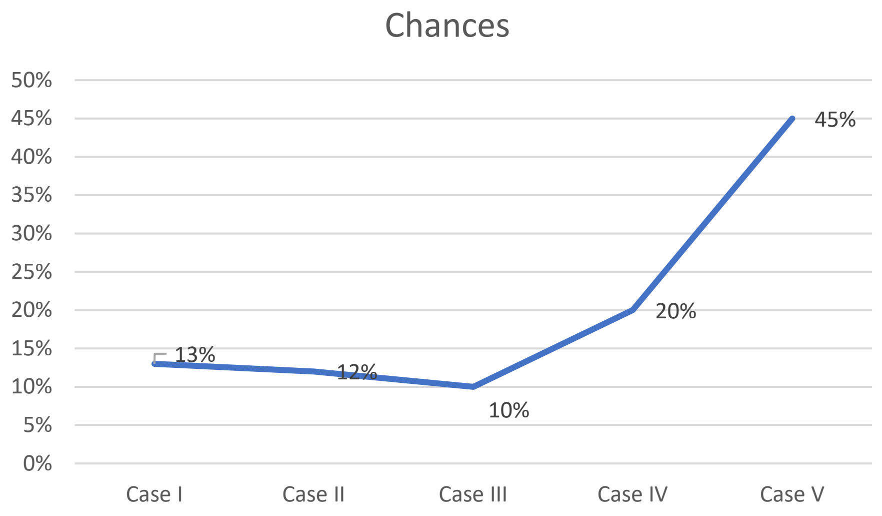


Figure 3: Trend lines for the chances of five cases happen

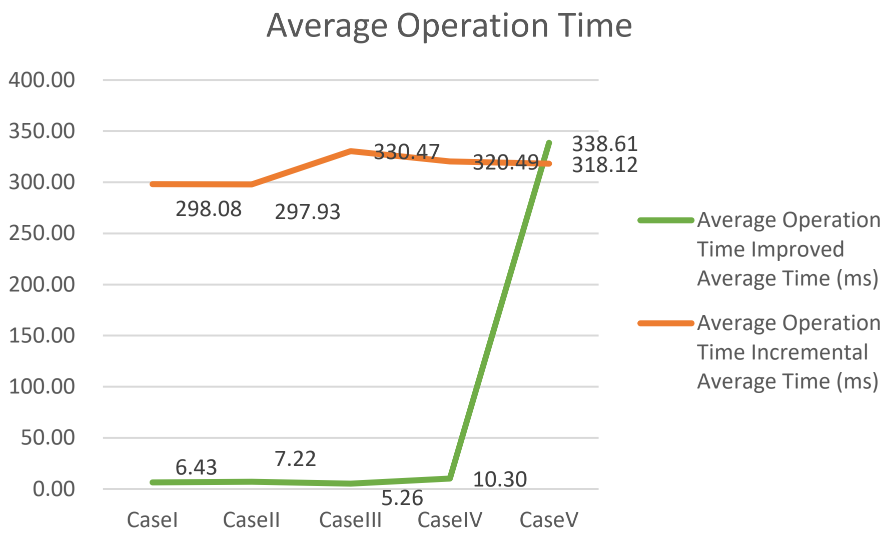


Figure 4: Trend lines for average process time for different cases

for each case in each dimension. The time unit is millisecond.

Table 2 shows the improvement percentage of improved IAA.

$$\text{Percentage} = \frac{\text{Improved IAA average time} - \text{IAA average time}}{\text{IAA average time}}$$

5 Performance Analysis

Table 1 shows the typical data collected from the experiments stated previously. The average operation time of the Incremental Assignment Algorithm (IAA) required by the random matrix is linearly increased. It is possible for a particular smaller dimension matrix to take more time than a larger dimension matrix because the value distributions significantly affect the time needed in the relevant algorithm. The average running time of IIAA is stable regardless of the increase of the dimension. The average running time difference between IIAA and IAA becomes higher because of the different numbers of iterations.

Table 2 indicates that with a higher dimension, the number of iterations of IAA will go higher. But IIAA still has only one iteration no matter how high the dimension is.

In Case I, Case II, and Case III, IIAA saves up to 96% of

Table 1: The process time for each case

Dimension / Case & Time	200 × 200 dimension	100 × 100 dimension
Case I		
Improved Average Time	7.74	5.13
Incremental Average Time	462.92	133.24
Case II		
Improved Average Time	9.75	4.71
Incremental Average Time	476.35	119.52
Case III		
Improved Average Time	6.67	4.47
Incremental Average Time	528.81	132.14
Case IV		
Improved Average Time	8.08	12.53
Incremental Average Time	519.46	121.51
Case V		
Improved Average Time	545.78	131.44
Incremental Average Time	515.84	120.41

Table 2: The percentage that the improved IAA improves

Dimension / Percentage	200 × 200 dimension	100 × 100 dimension
Case I	-98.32%	-96.15%
Case II	-97.95%	-96.03%
Case III	-98.74%	-96.62%
Case IV	-98.45%	-89.69%
Case V	5.80%	9.16%

operation time in 100 dimensions matrix and nearly 98% in 200 dimensions matrix. In Case IV, IIAA saves about 90% of operation time. The data express that the Improved Incremental Assignment Algorithm will save much more time than the Incremental Assignment Algorithm.

For all the 200 random matrices, 55% matrices meet the conditions of IIAA, the rest of the matrices will go to IAA that is the Case V (general case).

If the matrix is produced randomly, the general expectation of improvement is 46.5% compared with the operation time using the incremental assignment algorithm.

Overall, when the matrix is solved by IIAA, the processing time will be much faster than IAA.

6 Conclusion

In this paper, an improved algorithm has been proposed to solve the incremental assignment problem. From the view of the overall program, the algorithm improves a lot with the running time of operation. From finding the largest difference of the row and the column, with the use of exchange with its maximum weighted matching, the problem has been reduced. Consequently, the step of iteration can be reduced largely.

From the perspective of functions, our solution directly provides an improved way to solving the Incremental Assignment Problem. Not only the operation time will be saved, but also the occupied space will be reduced.

The computational complexity of our algorithm is $O(n^2)$, because the most complicated case for our algorithm will go to the incremental assignment algorithm. As the complexity of the incremental assignment algorithm is $O(n^2)$, our algorithms are also at the same level.

Our solution has many advantages, but it is still necessary to point out the disadvantages. About 55% of the 200 random matrices meet the conditions of the improved incremental assignment algorithm, the rest of the matrix will go to the incremental assignment algorithm. The chance is limited and needs to be improved in the future.

References

[1] R. E. Burkard, M. Dell'Amico, and S. Martello, "Assignment Problems," Society for Industrial and Applied Mathematics, pp. 35-144, 2009.

[2] https://en.wikipedia.org/wiki/Assignment_problem, 2019.

[3] H. W. Kuhn, "The Hungarian Method for the Assignment Problem," Naval Research Logistics Quarterly 2(1–2):83-97, March 1955.

[4] J. Munkres, "Algorithms for the Assignment and Transportation Problems," *Journal of the Society for Industrial and Applied Mathematics* 5(1):32-38, March 1957.

[5] I. H. Toroslu and G. Üçoluk, "Incremental Assignment Problem," *Information Sciences* 177(6):1523–29, March 15, 2007.

[6] I. H. Toroslu, and G. Üçoluk, Authors' Response to "An Addendum on the Incremental Assignment Problem," by Volgenant, Vol. 178, 2008.

[7] Y. Xie, "An o(n2.5) Algorithm: For Maximum Matchings in General Graphs," *Journal of Applied Mathematics and Physics* 6(9):1773-82, 2018.

Pinzhi Wang (photo not available) received the M.S. degree in computational sciences from Laurentian University, Sudbury, ON, Canada, in 2020. She is currently a Mathematics Teacher in Private School.

Youssou Gningue (photo not available) graduated with a Doctorate, Ph. D., in Applied Mathematics from the University of Sherbrooke (Quebec, Canada) in 1992. He has been a professor in the Mathematics and Computer Science Department of Laurentian University since September 1991 where he is cur-rently an emeritus full Professor since May 2021. Originally, from the country of Sénégal, Professor Youssou Gningue teaches at the third cycle of Applied Mathematics at the University Cheikh Anta Diop (UCAD), Dakar, Sénégal where he supervises doctoral students. Professor Youssou Gningue is also a Pan-Africanist who campaigns for the Unity of Africa

H**aibin Zhu** is a Full Professor and the Coordinator of the Computer Science Program, the Founding Director of Collaborative Systems Laboratory, a member of the University Budget Plan committee, Arts and Science Executive Committee, and the Research Committee, Nipissing University, Canada. He is also affiliate professor of Concordia Univ. and adjunct professor of Laurentian Univ., Canada. He received B.S. degree in computer engineering from the Institute of Engineering and Technology, China (1983), and M.S. (1988) and Ph.D. (1997) degrees in computer science from the National Univ. of Defense Technology (NUDT), China. He was the chair of the department of Computer Science and Mathematics, Nipissing University, Canada (2019-2021), a visiting professor and a special lecturer in the College of Computing Sciences, New Jersey Institute of Technology, USA (1999-2002) and a lecturer, an associate professor and a full professor at NUDT (1988-2000). He has accomplished (published or in press) over 200 research works including 29 IEEE Transactions articles, six books, five book chapters, three journal issues, and three conference proceedings. He is Senior member of ACM, a full member of Sigma Xi and a senior member of IEEE.