Sustainable Computing: Informatics and Systems 46 (2025) 101135

Contents lists available at ScienceDirect

Sustainable Computing: Informatics and Systems

journal homepage: www.elsevier.com/locate/suscom

A parallel computational approach for energy-efficient hydraulic analysis of water distribution networks using learning automata

Ali Suvizi, Ruhollah Ahmadi, Morteza Saheb Zamani [*], Mohammad Reza Meybodi

Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran

A R T I C L E I N F O

Keywords:
Parallel computing
Learning automata
Energy-efficiency
Taylor series
Sustainable water management
Steady-state hydraulic analysis
Water distribution network

A B S T R A C T

The hydraulic analysis of water distribution networks (WDNs) is crucial for ensuring efficient management of water resources, a key aspect of sustainable urban development. Formulation and steady-state hydraulic analysis of these networks have been conducted using both numerical and non-numerical methods. WDN hydraulic equations are complex and non-linear, requiring multiple executions, making their hydraulic analysis computationally demanding and energy intensive. This paper introduces an energy-efficient parallel computing approach using learning automata to significantly enhance the speed and energy efficiency of hydraulic analysis. By employing a cellular automaton framework that reflects the WDN structure, and a solution methodology based on the Taylor series enhanced with learning automata, we propose a system that reduces computational time and energy consumption. We compare the performance of our proposed approach with the EPANET software across networks of varying complexity and topologies. The results suggest our parallel algorithm not only accelerate the hydraulic analysis process up to 60 times compared to existing methods, but also significantly decrease the energy consumption, highlighting its potential for sustainable water management practices.

1. Introduction

Sustainable management of water distribution networks (WDNs) is increasingly recognized as pivotal for the ecological and societal well-being of communities. The steady-state analysis of WDNs plays a fundamental role in the design, optimization, leakage detection, and maintenance of these systems [1–3]. Over the past few decades, numerous algorithms have been developed to solve linear and non-linear equations associated with steady-state hydraulics [4]. Numerical modeling is one of the most effective algorithms for this analysis. However, these methods often face challenges due to uncertain modeling parameters, which make them complex, time-consuming and energy intensive. Additionally, many applications necessitate the analysis of WDNs' dynamic behavior over time, requiring a large number of times to solve the network equations at each step. Therefore, developing methods that can accelerate hydraulic analysis is crucial.

Numerous problems associated with WDNs can be addressed through steady-state hydraulic analysis, for which a variety of formulations have been proposed. Two major types of these formulations are local and simultaneous. The local approach addresses one equation at a time, continuously modifying flows to satisfy energy equations with acceptable accuracy, and is commonly used for simple networks due to its simplicity but is inefficient for larger networks [5]. In contrast, simultaneous approaches tackle multiple equations at once, such as using the Jacobian matrix for solving Q equations, and are used in commercial software like KYPIPE [6,7].

Newton-based Global Gradient Algorithm (GGA), introduced by Todini et al. [22], is a popular method for computing steady-state hydraulic variables. It determines the pressure in the nodes and the flow in the pipes by repeatedly solving a system of non-linear equations using the conjugate gradient method [8]. Commercial software like EPANET and WATERGEMS [9] utilize this algorithm for analyzing WDNs. Despite being the fastest method for such analysis, GGA becomes increasingly slow as the network and the population grow in size. The time-consuming matrix inversion process is the main issue with this method. Therefore, researchers have attempted to speed up the simulation by using clusters of personal computers (PC), multi-processor platforms, graphics processor units (GPUs), and field-programmable gate arrays (FPGA).

Guidolin et al. accelerated the numerical solution of hydraulic equations of WDNs using a single instruction, multiple data (SIMD) paradigm on a GPU, but no significant speed-up was achieved [10].

* Corresponding author.
E-mail address: szamani@aut.ac.ir (M. Saheb Zamani).

https://doi.org/10.1016/j.suscom.2025.101135
Received 25 March 2024; Received in revised form 15 February 2025; Accepted 5 May 2025
Available online 10 May 2025

Peinado et al. focused on parallelizing matrix-vector multiplication to reduce the execution time of the CG algorithm, which only improved the speed of parallel multiplication operations [11,12]. Diao et al. presented a systematic approach using the Schür domain decomposition method to significantly reduce the computational complexity of a system, improving hydraulic modeling efficiency [13,14]. Ivetić et al. investigated the feasibility of accelerating a hydraulic engine by simultaneously solving simplified loop flow equations using the ΔQ method, demonstrating its efficiency in WDN optimization [15]. Vasilic et al. proposed improving the ΔQ method with a new TRIangulation BAsed Loops (TRIBAL) identification algorithm, which performs fewer computations over GGA [16].

Suvizi et al. [17] proposed a parallel computational architecture named "CA_Linear_RTL" rooted in the concept of CA to accelerate the numerical solution for steady-state hydraulic analysis of WDN. In their study, a WDN is divided into several independent cells, following the rules of CA neighborhoods. Subsequently, the non-linear hydraulic equations for these cells are formulated based on H equations and linearized in parallel using a Taylor series expansion. Each cell performs a series of small, finite calculations simultaneously. This parallel CA-based architecture is implemented on FPGAs, as CA-based algorithms are notably compatible with hardware platforms that feature parallel architecture.

This paper proposes a novel approach to accelerate the hydraulic analysis of WDNs by utilizing a parallel architecture using learning automata (LA). This method modifies the traditional hydraulic analysis to enhance its energy-efficiency and reduce computation time. The WDN is divided into small sub-networks called cells, each consisting of a central node and its neighboring nodes. Mass and energy conservation laws are simultaneously solved for each central node and its neighbors. This process is facilitated by applying a set of rules based on the current state of each cell and its neighboring cells [18]. The proposed algorithm combines cellular automata (CA) and LA. CA allows each cell to behave independently, determining its next state based on a set of rules and the current states of neighboring cells. Conversely, LA introduces a learning capability to adapt and improve decision-making within each cell. By incorporating LA in each CA cell, the hydraulic analysis of WDNs can be performed more efficiently.

The final step to solve the non-linear hydraulic equations of WDN involves the linearization method with the Taylor series. This method linearizes the set of non-linear equations, enabling parallel and efficient computation. By integrating LA and CA with the parallel computation process and the linearization method, this paper proposes a significant step towards integrating sustainability into the computational analysis of water distribution systems.

The structure of this paper is as follows: An introduction to the fundamental concepts of WDNs is presented in Section II. Section III details our proposed approach for speeding up the solution of equations in hydraulic system analysis. The methodology is then exemplified in Section IV. The description of the experimental results and application of the method to a WDN is discussed in section V. Finally, Section VI draws the primary conclusions from our research and suggests possible future developments.

2. Material and methods

2.1. Model of WDN analysis

Link-node model is a widely used and efficient method for representing WDNs. In this model, a node consists of junctions, reservoirs, and tanks. Junctions are the places where water is consumed and reservoirs are unlimited sources of water like groundwater or lakes, while tanks can store water with a limited capacity. A link represents pipes, pumps, or control valves. This paper focuses on networks comprising pipes, junctions, and reservoir to show the main concept, as many WDNs are in this format. This model calculates pressure heads and flows at nodes and links using the laws of mass and energy conservation, respectively [19]. The crucial components of this model are detailed below [20,21].

2.1.1. Head

The hydraulic head refers to the equivalent water height corresponding to a specific pressure exerted by the water column at the base of the container. Water pressure in WDNs refers to the force exerted by the water within the distribution system's pipes.

2.1.2. Head-loss

In pipes, head loss occurs due to friction between the flow and the pipe wall. Head loss in water pipes can be modeled with the Hazen-Williams model. Eq. (1) represents the Hazen-Williams head-loss relationships.

$$h_{f_j} = \frac{10.67 * L_j * Q_j^{1.852}}{C_j^{1.852} * d_j^{4.87}} \tag{1}$$

In this equation, for a pipe j, C_j is the stiffness coefficient, d_j is its diameter, L_j is its length and Q_j is the water flow in the pipe.

2.1.3. Mass and energy conservation

According to the law of mass conservation in water distribution networks, the algebraic sum of input and output flow of water to node j must be zero. Eq. (2) represents the law of mass conservation:

$$\sum_k Q_k = q_j \tag{2}$$

In this equation, q_j represents the demand of node j, and Q_k represents the flow of each node k which is connected to node j. The law of conservation of energy is represented by Eq. (3).

$$h_k = H_i - H_j = r_k * Q_k{}^n \tag{3}$$

In this equation, H_i and H_j are the head values at the beginning and end of the pipe k, r_k is the resistance coefficient of pipe k, $n = 1.852$, and Q_k is the flow of pipe k. To ensure energy consistency across the network, particularly in loops where multiple paths connect two nodes, the loop energy conservation equation is also considered.

$$\sum_{i \epsilon loop} h_{f,i} = 0 \tag{4}$$

In this equation, $h_{f,i}$ represents the head loss in each pipe segment within the loop, ensuring that the total energy around any closed loop is conserved.

2.2. Hydraulic analysis of WDN

Hydraulic analysis of water distribution networks is performed using two methods: demand-driven analysis and pressure-driven analysis. Demand-driven model is a classical problem of WDS analysis, assuming that all nodal outflows are equal to the required demands regardless of the actual nodal pressures. This model is commonly used and integrated into various software tools, such as EPANET [22]. Consequently, this study focuses on water network analysis using this model. In this model, the water requirements of the nodes are fully satisfied, regardless of the pressure levels at these nodes. Consider a WDN including n_p pipes and n_t nodes with n_r fixed-head nodes. In this network, the flow in the n_p pipes and the heads of the $n_n = n_t - n_r$ nodes are unknown. By applying the laws of mass and energy conservation, a non-linear system represented by Eq. (5) can be derived [23]:

$$\begin{bmatrix} A_{11} & \vdots & A_{12} \\ \cdots & \cdots & \cdots \\ A_{21} & \vdots & 0 \end{bmatrix} \begin{bmatrix} Q \\ \cdots \\ H \end{bmatrix} = \begin{bmatrix} -A_{10}H_0 \\ \cdots \\ -q \end{bmatrix} \tag{5}$$

In this equation, $Q^T = \begin{bmatrix} Q_1, & Q_2, \cdots, & Q_{n_p} \end{bmatrix}$ is the pipe flow vector,

$H^T = [H_1, \quad H_2, \cdots, \quad H_{n_n}]$ is the unknown pressure head vector, $q^T = [q_1, \quad q_2, \cdots, \quad q_{n_n}]$ is the set of nodal demands that are not dependent on the nodal head and $H_0{}^T = [H_{n+1}, \quad H_{n+2}, \cdots, \quad H_{n_t}]$ is the known heads vector. A_{12} and A_{10} are called edge-to-non-source-node and edge-to-source-node connectivity matrices, respectively. In these matrices, each entry a[i,j] is set as:

$$a[i,j] = \begin{cases} 1 & \text{if pipe i leaves node j} \\ 0 & \text{if pipe i is not connected to node j} \\ -1 & \text{if pipe i enters node j} \end{cases} \tag{6}$$

A_{21} is also obtained from the equation $A_{21} = A^T{}_{12}$.

Matrix A_{11} is a diagonal matrix whose elements are obtained from Eq. (7).

$$A_{11} = \begin{bmatrix} k|Q_1|^{n-1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & k|Q_{np}|^{n-1} \end{bmatrix} \tag{7}$$

In this equation, n_p is equal to the number of pipes in the network for which the required flow rates are unknown, $n = 1.852$, and k is the resistance coefficient in the Hazen-Williams model. One of the most used methods for solving hydraulic network equations in WDNs is the GGA [22]. GGA creates a system of non-linear equations represented by a positive definite matrix. This method analyzes the non-linear equation system and efficiently determines the pressure and flow values at nodes and pipes, respectively. It is an iterative algorithm that will execute and calculate the head and flow values until achieving negligible change in values between iterations. The number of iterations may be very high, especially in complex networks. Moreover, matrix inversion may play a major role in the run time of the algorithm leading to high time complexity in each iteration.

2.3. Taylor series

Utilizing Taylor expansions is one of the most efficient methods to linearize nonlinear equations. Taylor's expansion results in a linear relationship when we take the first two terms of the series into account. Eq. (8) presents Taylor's expansion in a linear format, while Eq. (9) demonstrates the calculation of '*x*' using this expansion.

$$f(x) = f(x_0) + f^{(1)}(x_0)(x - x_0) \tag{8}$$

$$if\ f(x) = 0 \rightarrow x = x_0 - \frac{f(x_0)}{f^{(1)}(x_0)} \tag{9}$$

where $f^{(1)}(x_0)$ is the first derivative of f at x_0.

2.4. Cellular and learning automata

2.4.1. Cellular automata

John von Neumann conducted the first studies on CA in the late 1940s, motivated by biology [24]. Each individual cell comprises a finite state machine and a set of neighbors defined in relation to it. The neighborhood of a two-dimensional array can be defined in various ways. The most well-known definitions are Moore's neighborhood and Neumann's neighborhood. In the Moore's neighborhood, all the eight cells adjacent to the central cell are considered as the neighborhood. However, the Neumann neighborhood includes only the top, bottom, left, and right cells [25]. The common CA neighborhoods are illustrated in Fig. 1.

To model a system using CA, four essential components are needed: a cell, the cell's state, the cell's neighborhood, and an updating rule that changes the cell's state in each iteration. CA are characterized by their parallelism, display representation, and homogeneous local structure. Parallelism implies that the states of all the cells change simultaneously with each update occurring in parallel. The local structure of CA means that a cell's new state is determined solely by its previous state and the states of its neighbors. The homogeneity feature involves rule-based updating of all cells [26,27]. There are generally two types of updating rules: ad hoc and analytical. Various updating rules have been explored in relation to optimization problems. Experimental results suggest that CA models using ad hoc updating rules are typically more flexible. In contrast, CA models employing analytical rules are faster and more suitable for computationally time-intensive situations. [28].

2.4.2. Learning automata

A learning automaton, an adaptive decision-making unit [29,30], enhances its performance by learning to choose the best action from a finite set of alternatives in a random environment. Each action in the set is assigned a probability distribution, and the chosen action is given as input to the environment. In response to the selected action, the environment provides a reinforcement signal. The probability vector of the action is adjusted according to this reinforcement feedback from the environment. A learning automaton aims to identify the optimal action among the set to minimize the average penalty it receives from the environment.

The environment can be described by a triple $E \equiv \{\alpha, \beta, c\}$, where $\alpha \equiv \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ denotes the set of inputs, $\beta \equiv \{\beta_1, \beta_2, \ldots, \beta_m\}$ denotes the set of values that the reinforcement signal can take, and $c \equiv \{c_1, c_2, \ldots, c_m\}$ represents the penalty probability set, with each element ci representing the given action α_i.

An environment is referred to as *stationary random* if the penalty probabilities remain consistent. Conversely, an environment that changes over time is known as *non-stationary* environment. Environments can be categorized into P-model, Q-model, and S-model based on the reinforcement signal β. A P-model environment only accepts two binary values: 0 and 1. In the Q-model environments, the reinforcement signal can assume a finite number of values within the range of [0,1]. S-model environments allow any reinforcement signals within the range of [a, b]. Fig. 2 illustrates the learning automaton within its random environment.

The learning automata are divided into two main categories [31]: fixed structure learning automata (FSLA) and variable structure learning automata (VSLA). There are three components of VSLA $\langle \beta, \alpha, T \rangle$, where β is the set of inputs, α is the set of actions, and T is the learning algorithm.

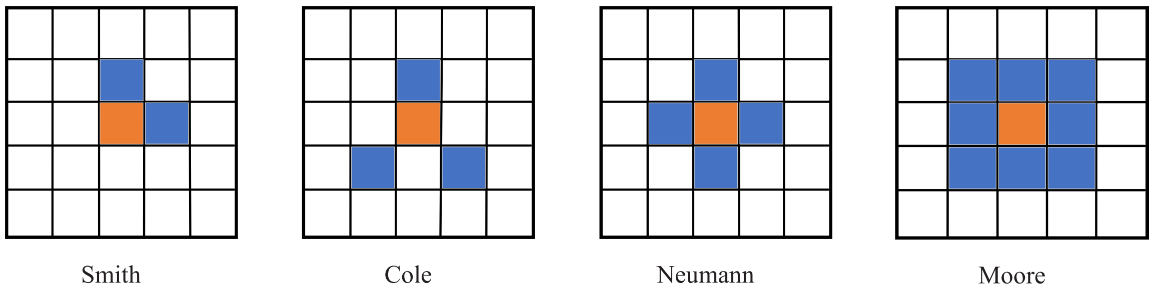


Fig. 1. Common neighborhood structures in cellular automata.

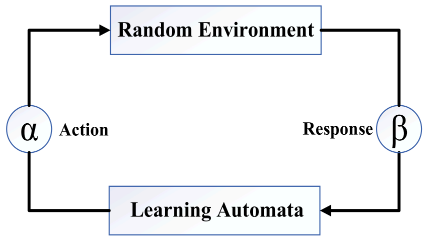


Fig. 2. Learning Automata in a random environment.

The learning algorithm modifies the action probability vector by using a recurrence relation. Let *α(k)* and *p(k)* indicate the selected action at instant *k* and the probability vector based on which it is selected, respectively. A linear learning algorithm updates the action probability vector *p* using the recurrence equation shown in (10) and (11). Let α(k) be the action chosen by the automaton at instant *k*.

$$p_j(k+1) = \begin{cases} p_j(k) + a\left[1 - p_j(k)\right] & j = i \\ (1-a)p_j(k) & \forall j \neq i \end{cases} \quad \textit{if action gets reward} \tag{10}$$

$$p_j(k+1) = \begin{cases} (1-b)p_j(k) & j = i \\ \left(\frac{b}{r-1}\right) + (1-b)p_j(k) & \forall j \neq i \end{cases} \quad \textit{if action gets punish} \tag{11}$$

An automaton can choose up to *r* actions, and *a(k)* and *b(k)* are reward and penalty parameters that determine how much the probability of an action increases and decreases. If *a(k)*= *b(k)*, the recurrence Eqs. (10) and (11) are called the linear reward-penalty (LRP) algorithm. If *a(k)* = *εb(k)* where 0< ε<1, then the learning algorithm is called the linear reward-ε-penalty (LRεP). Finally, if *b(k)* = *0*, the equations are called linear reward-inaction (LRI).

3. Theory and calculation

The complexity and nonlinearity in the head-flow equations of WDNs not only increase the modeling execution time but also amplify energy consumption, thereby reducing modeling accuracy and efficiency. Recognizing these challenges, this section introduces a parallel algorithm that uses CA and LA for the numerical solution of the hydraulic model, aiming to solve WDN difficulties efficiently.

The proposed method employs a demand-driven model, which remains highly relevant for systems where the pressure typically suffices to meet demands. This approach is particularly advantageous in scenarios where an initial analysis is needed to identify potential pressure inadequacies before a more detailed pressure-driven study is conducted. By focusing on demand-driven analysis, our model streamlines the computational process, enabling a clearer focus on enhancing computational efficiency and energy savings.

Each WDN node in our model is considered a cell, with the head of nodes defined as the cell state. Applying Neumann's neighborhood rule to a cell simplifies the complex and non-linear equations related to the hydraulic model of the water distribution system, thus accelerating the equation-solving processes. The environment of the proposed algorithm is based on the P-model, where the reinforcement signal β can only be zero or one. According to the updating rules, a probability vector with ten entries ranging from zero to one is assigned to each cell. The length and values of this vector are finely tuned for optimal balance between time and accuracy considerations, with initial values set at *1/n*, which are then updated based on the LR∈P algorithm in accordance with Eqs. (10) and (11).

By using this demand-driven approach, we are able to maintain simplicity and focus on computational efficiency and energy savings without the added complexity of pressure-responsive modeling. This foundational method not only provides a basis for immediate application in suitable scenarios but also sets the stage for future extensions to include pressure-driven analysis, thereby broadening the applicability and enhancing the realism of our simulations.

Summary of contribution:

- ➢ Hydraulic Equations Linearization: The method applies Taylor series expansions to linearize the non-linear hydraulic equations of the WDN. This mathematical approach simplifies the equations, making them easier and faster to solve.
- ➢ CA for Network Segmentation: The network is divided into segments called cells, each representing a node and its immediate connections. This segmentation allows for parallel processing where each cell calculates its state independently, enhancing computational speed.
- ➢ Writing and Solving Mass Conservation Equations: Mass conservation equations are written for each node and solved simultaneously across the network, ensuring that the inflows and outflows at each node are balanced. This process is crucial for maintaining the physical accuracy of the network model.
- ➢ Flow Calculation and Convergence Analysis: At the end of each iteration, flow rates in each pipe are calculated, and the mass conservation at each node is checked. Energy conservation is also monitored to assess the convergence of the system towards a steady state, demonstrating the effectiveness of the model in simulating real-world hydraulic conditions.
- ➢ LA for Iterative Optimization: LA is employed to dynamically adjust the calculations within each cell based on feedback. This adaptive mechanism not only optimizes convergence and accuracy but also significantly reduces the number of iterations needed to reach a solution.

3.1. Equations of a cell

A water distribution network is taken as input and divided into n_n cells, where n_n represents the number of non-reservoir nodes. Following the Neumann's neighborhood rule, each cell comprises a central node and a minimum of one to a maximum of four adjacent nodes associated with the neighboring pipes of the central node. These cells can be processed simultaneously to calculate the node heads and pipe flows. Pertaining to this, the mass conservation equations for the nodes are derived based on Eq. (12). In contrast, the head-loss equations for each pipe and between the two interconnected nodes are derived based on Eq. (13).

$$\sum_{l\epsilon j_{in}} Q_l^m - \sum_{l\epsilon j_{out}} Q_l^m - q_i = 0 \quad \forall i \in (1, n_n) \tag{12}$$

$$H_j^m - H_i^m = K_l \left|Q_l^m\right|^n \rightarrow \quad Q_l^m = sign\left|\frac{H_j^m - H_i^m}{k_l}\right|^{\frac{1}{n}} \tag{13}$$

In these equations, n_n is the number of nodes, H_i^m and H_j^m denote the head values for the two ends of the pipe between nodes *i* and *j* in iteration *m*, respectively, Q_l^m represents the flow rate of pipe *l*, k_l stands for the resistance coefficient of pipe *l*, q_i represents the demand of node *i*, j_{in} is the set of inlet flow pipes, j_{out} represents the set of outlet flow pipes, and $n = 1.852$.

According to the proposed computational architecture in [17], which is based on the CA concept used in solving WDNs and by adding the LA principle, the next state of a central node in each cell is determined by its current state and the states of its neighboring nodes. In our algorithm, the state of a cell is considered the 'head' of a node, and its value is derived from the current 'head' values of the neighboring nodes. Consequently, while we know the 'head' of the neighboring nodes and the demand of the central node, the 'head' of the central node and the pipe flow remain unknown. The 'heads' of the central nodes are calculated iteratively and simultaneously using the cells' non-linear Eqs. (12), (13).

3.2. Parallel calculation of WDNs

This stage involves formulating the equilibrium equations of mass at the node and energy equilibrium in the pipes in terms of heads, which simplifies the calculations by reducing the number of unknowns. This approach enables easier and faster equation-solving. Based on the proposed algorithm, the WDN is segmented into several cells according to the CA. Each cell operates on the LA logic. Taking into account the environment, this logic facilitates the updating of the probability vector, action selection, and the receipt of the environment response signal. Fig. 3 illustrates a cell following Neumann's neighborhood rule, which consists of a central node and its neighbors.

In this figure, the central node is N1. The neighboring nodes with known heads are N2, N3, N4, and N5. By utilizing Eqs. (12) and (13), we can formulate the mass conservation equation for the central node. Eqs. (14) to (17) are the head-loss equations in the pipes between the node pairs.

$$Q_2 = sign\left|\frac{H_2 - H_1}{k_2}\right|^{\frac{1}{n}} \tag{14}$$

$$Q_3 = sign\left|\frac{H_3 - H_1}{k_3}\right|^{\frac{1}{n}} \tag{15}$$

$$Q_4 = sign\left|\frac{H_4 - H_1}{k_4}\right|^{\frac{1}{n}} \tag{16}$$

$$Q_5 = sign\left|\frac{H_5 - H_1}{k_5}\right|^{\frac{1}{n}} \tag{17}$$

Eq. (18) states that the sum of the input and output flows to and from node N1 is proportional to the flow's direction in each pipe. It shows the mass conservation equation for the central node.

$$Q_2^m + Q_5^m + Q_3^m + Q_4^m - q_1 = 0 \tag{18}$$

The final equation for the central node will have one unknown variable remaining after substituting the head-loss equations of pipes into Eq. (18). The flow's direction is taken into account when determining the sign between expressions in this context.

$$\begin{aligned} sign\left(\frac{H_2^m - H_1^m}{k_2}\right)^{\frac{1}{n}} + sign\left(\frac{H_5^m - H_1^m}{k_5}\right)^{\frac{1}{n}} + sign\left(\frac{H_3^m - H_1^m}{k_3}\right)^{\frac{1}{n}} \\ + sign\left(\frac{H_4^m - H_1^m}{k_4}\right)^{\frac{1}{n}} - \quad q_1 \quad = 0 \end{aligned} \tag{19}$$

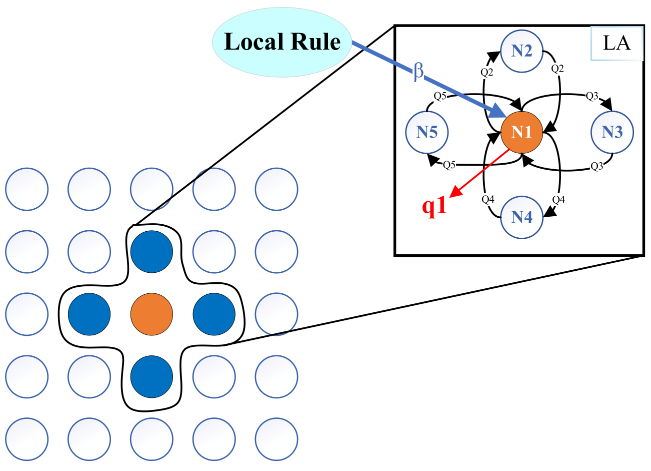


Fig. 3. A cell based on the Neumann's neighborhood.

This equation will be written for all cells (central node and its neighbors) using CA, and by using LA and Taylor series, it will be solved simultaneously. By solving this nonlinear equation using the Taylor series and the head values from the previous iteration, the head value of each cell (H_i^{m+1}), which represents the head of node i in the iteration, $m+1$ is obtained.

3.3. Linearizing with Taylor series expansion

The hydraulic equations were reduced to one equation (Eq. 19) for each node in the previous section. This is a nonlinear equation, which is linearized using the Taylor series based on Eq. (9). The only unknown value in the equation (H_i^{m+1}) is determined as follows:

$$\begin{aligned} F_{H_1^m} =& sign\left(\frac{H_2^m - H_1^m}{k_2}\right)^{\frac{1}{n}} + sign\left(\frac{H_5^m - H_1^m}{k_5}\right)^{\frac{1}{n}} + sign\left(\frac{H_3^m - H_1^m}{k_3}\right)^{\frac{1}{n}} \\ & + sign\left(\frac{H_4^m - H_1^m}{k_4}\right)^{\frac{1}{n}} - q_1 \end{aligned} \tag{20}$$

$$\begin{aligned} DF_{H_1^m} =& \left(-\frac{1}{n}\right) * \left[\left|\frac{H_2^m - H_1^m}{k_2}\right|^{\frac{1}{n}-1} + \left|\frac{H_5^m - H_1^m}{k_5}\right|^{\frac{1}{n}-1} + \left|\frac{H_3^m - H_1^m}{k_3}\right|^{\frac{1}{n}-1} \right. \\ & \left. + \left|\frac{H_4^m - H_1^m}{k_4}\right|^{\frac{1}{n}-1}\right] \end{aligned} \tag{21}$$

$$H_1^{m+1} = H_1^m - w * \left(\frac{F_{H_1^m}}{DF_{H_1^m}}\right) \tag{22}$$

In this equation, H_1^m is the value of the central node in the previous iteration, and H_1^{m+1} is the value of the central node in the current iteration. In the following, α is briefly used in the equations instead of $\frac{1}{n}$.

3.4. Steps for solving hydraulic equations

The approach to solving the hydraulic equations starts with each central node in cells of WDN being associated with Eq. (19), which integrates potential differences with neighboring nodes. These differences drive the flow between nodes under hydraulic gradients. Due to the nonlinear nature of the equation, a Taylor series expansion is employed for linearization, centered around the head value from the previous iteration H_i^m. This simplifies the nonlinear equation into a linear form that approximates the behavior of the nonlinear system under small changes in head.

Subsequently, the derivatives of the nonlinear function with respect to the central node head values, shown in Eq. (21) as derivative $DF_{H_i^m}$ are calculated. This equation is essential for understanding how changes in one node's head affect another and facilitates the application of iterative solvers. Using the previous iteration's head values, the linearized system is solved iteratively. The update Eq. (22) adjusts the head based on the gradient of the solution landscape, defined by the $F_{H_i^m}$ and $DF_{H_i^m}$, and a step size w, progressively moving towards the system's equilibrium state.

At the end of each iteration, the change in head values (ΔH) is calculated. This difference is used to check for convergence; if (ΔH) falls below a predetermined threshold, the solution is considered stable, and the iteration process is halted.

3.4.1. Integration of dynamic hydraulic components

In the pursuit of refining the simulation of WDNs, our study extends beyond the static calculations of the network hydraulics to incorporate dynamic elements such as pumps and tanks, which are pivotal in real-world applications. This section builds upon the foundational methodologies discussed earlier, particularly focusing on how these components are seamlessly integrated into our parallel computation framework for WDNs and its limitations.

Dynamic hydraulic components like pumps and tanks introduce variable behaviors based on operational conditions and time, necessitating a more detailed approach to accurately model their impact within a demand-driven analysis framework. Pumps, which enhance hydraulic head and thus influence flow rates across the network, and tanks, which serve as reservoirs with fluctuating water levels affecting the network pressure, are crucial for the realism and applicability of hydraulic models.

Given the complexity and the dynamic nature of these components, this section will delve into the mathematical formulations used to represent their functionality. It will further discuss the integration of these formulations into the existing parallel computational architecture, emphasizing the modifications necessary to accommodate the non-static characteristics of pumps and tanks.

Our current research primarily emphasizes steady-state simulations to validate the parallel computational architecture we've developed. Although our model's structure—featuring modular design, cell independence, and learning capabilities—is well-suited for dynamic conditions like those found in Extended Period Simulations (EPS), we have initially focused on steady-state conditions. This focus aimed to establish a robust and fast computational system before exploring the complex dynamics of EPS. Moving forward, we plan to expand our methodology to include EPS, which will allow us to leverage the full dynamic potential of our system for simulating real-time network behaviors over extended periods. This expansion is expected to significantly enhance the scope and utility of our model, as outlined in the discussions of current limitations and future research directions.

Tank: To incorporate a tank into our existing hydraulic model equations, it is needed to treat it as a node with a variable head that depends on the stored water volume. Eq. (23) shows its head calculation.

$$H_{tank}{}^{m} = H\_elevation + \frac{V_{water}{}^{m}}{A} \tag{23}$$

In the given equation, $H_{elevation}$ is the initial head of tank, V_{water} is the volume of water in the tank, and A is the cross-sectional area of the tank. In order to incorporate this equation into our hydraulic equations, we must write the mass conservation equation for each tank, similar to all other nodes. Using the head-loss equation, we then rewrite the mass conservation equation in terms of heads. The mass conservation equation for the tank is shown in Eq. (24).

$$\sum_{i=neighbor\ nodes} sign\left|\frac{H_i^m - H_{tank}^m}{k_l}\right|^{\frac{1}{n}} - q = 0 \tag{24}$$

In this equation, H_{tank}^m is the tank head in the m^{th} iteration, H_i^m is the head value of its neighboring node, q is demand and k_l stands for the resistance coefficient of the connected pipe, with $n = 1.852$. In each iteration, after head calculation for junctions, we need to calculate flow of pipes, especially those connected to the tank and use that flow to update tank volume based on Eq. (25). This updated tank volume will then be used to update the tank head using Eq. (23) for the next iteration.

$$V^{(2)} = V^{(1)} + (Q_{in} - Q_{out}).\Delta t \tag{25}$$

In the given equation, V is the volume of water in the tank and Q_{in} and Q_{out} are net flow into or out of the tank, respectively.

Pump: To incorporate a pump into our existing hydraulic equations for WDNs, it is need to account for the way a pump influences the hydraulic head between its inlet and outlet. A pump adds a specific head increase (ΔH_{pump}) that is typically dependent on the flow rate passing through the pump. This relationship can be defined using a pump curve, which describes how the head increase varies with flow.

More commonly, pumps are treated as links between two nodes. In this case, the head at the downstream node is the head at the upstream node plus the head increase provided by the pump. To incorporate a pump into the existing hydraulic balance equation, we can modify the flow equations between nodes where the pump is installed. Assuming the pump is between nodes i and j, the modified equation for node i is shown in Eq. (26).

$$\sum_{k=junctions}\left(sign\left|\frac{H_k^m - H_i^m}{k_l}\right|^{\frac{1}{n}}\right) + \left(sign\left|\frac{(H_j^m + \Delta H_{pump}) - H_i^m}{k_l}\right|^{\frac{1}{n}}\right) - (q_i) = 0 \tag{26}$$

In this equation, ΔH_{pump} is the head increase due to the pump, which can be calculated from the pump curve based on the flow rate Q and H_j^m is the hydraulic head at the pump's inlet node in the m^{th} iteration. The first element of this equation represents the main mass conservation for node i with all its neighbors excluding the node that has a pump between it and node i. The second element is the equation for nodes i and j that have a pump between them. In each iteration of computation, it is needed to update the ΔH_{pump} based on the flow rate calculated in that iteration. Eq. (27) shows how to perform this update by the end of each iteration.

$$\Delta H_{pump} = f(Q) = a - bQ - cQ^2 \tag{27}$$

In this equation a, b *and* c are coefficients determined by the pump's characteristics and Q is the flow rate through the pump. By adjusting these coefficients, we can calibrate the pump curve.

3.5. LA calculation in CA's cell

In complex systems with uncertainty and limited information, LA introduces a specific approach for reinforcement learning. LAs are stochastic models that make decisions based on past experiences and interactions with the environment.

The learning algorithm is formulated as follows [32]:

1. Initialization: Initialize the probabilities of selecting actions. A vector typically represents these probabilities.
2. Action Selection: Select an action based on the current probabilities.
3. Environment Interaction: Interact with the environment by executing the selected action and receiving a punishment/reward signal.
4. Probability Update: Update the probabilities based on the received reinforcement signal. The updating procedure uses a learning rule that adjusts the probabilities towards higher rewards.
5. Repeat Steps 2–4 until convergence or a predefined stopping criterion is met.

The formulation of the learning algorithm in CA for calculating the head of nodes can be described as follows:

1. Neighborhood Selection: Select a neighborhood for each node. The neighborhood defines the set of neighboring nodes that influence the value of the current node.
2. Value Calculation: Calculate the new value for each node based on the values of its neighboring nodes. In the learning algorithm, the value calculation is performed using a function that combines the values of the neighboring nodes.
3. Update: Update the values of the nodes based on the calculated values in step 2. The update is typically performed simultaneously for all nodes in the cellular automaton.
4. Repeat Steps 1–3 until convergence or a predefined stopping criterion is met.

The objective of the learning algorithm in CA is to learn the optimal values for each node in the automaton. The algorithm aims to capture the relationships and dependencies between nodes by considering the values of the neighboring nodes in the value calculation process in each iteration.

This formulation leverages the principles of CA, where nodes interact locally with their neighbors to determine their own state or value. The learning algorithm in each cell introduces a function that combines the values of the neighboring nodes to calculate the new value for each node. In addition, a weighted parameter (w) is considered for updating the value of the heads in Eq. (22).

The concept of "weight" refers to the value assigned to each neighboring node based on the value difference between the current node and its neighbors. Using weights allows the algorithm to consider the importance or influence of each neighbor's value when calculating a node's new value.

LA is utilized to calculate the optimal value for the weight parameter (w) in each cell, using an algorithm that selects actions based on past experiences from the environment [33].

The proposed algorithm defines the updating rule based on Eq. (28).

$$H_n - H_{n-1} < \varepsilon \tag{28}$$

In the given equation, H_n and H_{n-1} represent the node's head values in the last and penultimate iterations, respectively. The equation illustrates the difference between the node's value in the n^{th} iteration and its value in the *(n-1)*th iteration, to be less than a certain threshold (ε). A selected action is rewarded if the parameter values for this condition decrease with each iteration relative to the prior one. This implies that the head value in each cell remains nearly constant across two consecutive iterations. Conversely, if the condition is not met, the selected action is penalized.

The learning automaton receives updated action probabilities as input and generates the weight parameter (w) as output. It is assumed that the value of (w) is chosen based on a random number generation procedure f ($f \in [0,1]$). With Eq. (29) as a constraint, the algorithm identifies a w that has a cumulative probability equal to or exceeding the generated random number.

$$\sum_{j=1}^{i} P_j \geq f \tag{29}$$

This iterative process is repeated until the convergence condition is met. The convergence condition is achieved when one entry in each probability vector exceeds 0.9. The pseudocode for the described method is presented in Algorithm 1.

Algorithm 1. LA in CA

```
[Input]: Parallel cells, heads, and pipe characteristics, stopping criterion $\boldsymbol{\varepsilon}$
[Output]: H

1.  While ($\boldsymbol{at\ least\ one\ entry\ in\ each\ probablity\ vector\ exceeds\ 0.9}$)
2.      For each cell $j$ with $n_{pj}$ neighbors
3.          For each neighbor $i$ of cell $j$
4.              $\boldsymbol{F_j = [\frac{\sum_{i=0}^{np_j-1} sign(H_i^m - H_j^m)}{k_{ij}}]^{\alpha} - q_j}$
5.              $\boldsymbol{DF_j = (-\alpha) * \left[\frac{\sum_{i=0}^{np_j-1} |H_i^m - H_j^m|}{k_{ij}}\right]^{\alpha-1}}$
6.          End for
7.          if (iteration! = 0)
8.              $\boldsymbol{[LA_{\Delta H(j)}]^T = [H^{(m+1)}]^T - [H^{(m)}]^T}$
9.              $\boldsymbol{if(|LA_{\Delta H(j)}| < \varepsilon)}$
10.                 Receive_Reward_Signal_from_Environment
11.             $\boldsymbol{else}$
12.                 Receive_Punish_Signal_from_Environment

13.         W = Choose_action ()
14.         $\boldsymbol{H_j^{m+1} = H_j^m - w * (\frac{F_j}{DF_j})}$
15.         Update $H_j$'s in all cells ($j \in [1, n]$)
16.     End for
17. End while
```

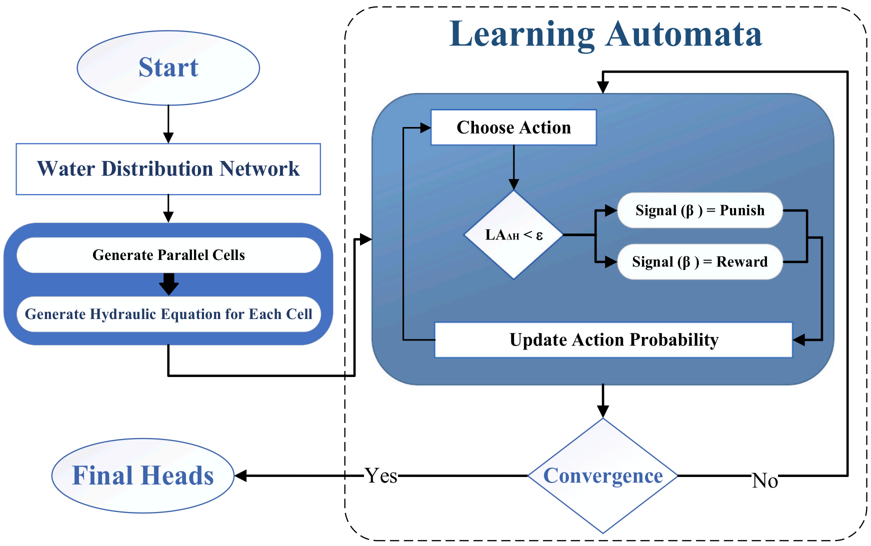


Fig. 4. The flowchart of the proposed algorithm for hydraulic analysis using LA.

Fig. 4 outlines the steps of the proposed algorithm. The system's input is the characteristics of the WDN, and the nodes' head values are generated as output. Initially, the network is partitioned into j cells, each representing a node with an unknown head value, based on Neumann's CA neighborhood. Then for each cell, the corresponding hydraulic equation and a unique LA are specified.

Each LA uses the head equations from each concurrent cell in the hydraulic analysis stage. An action representing the (w) parameter is chosen for every cell. The difference between the head value in the current and previous iterations is then calculated and compared with ε. A reward is issued if the difference is less than ε; otherwise, a penalty is given as a reinforcement signal β.

Subsequently, the action probability is updated based on β. The value of a successful action increases while the value of other actions decreases. This process repeats until all cells converge. Convergence is achieved when at least one entry in each probability vector reaches 0.9. Upon meeting this convergence condition, the final head values are reported.

In this architecture, each node, along with its four neighbors, forms a cell, and the hydraulic equations for all nodes are solved concurrently using the Taylor series, enabling each cell to adjust its decision-making process based on local conditions. After making a decision, each cell evaluates the result and measures its performance against predefined goals or optimization criteria. The cell subsequently updates its internal parameters, such as decision probabilities, based on the feedback received and its decision's success (reward) or failure (punishment). This decision-making, interaction with neighbors, and learning process are iteratively repeated for each cell. The cells modify their decision-making strategies and parameters through this iterative process, continuously refining the WDN hydraulic analysis. Fig. 5 illustrates the general structure of the proposed architecture, showing its components. Each component, numbered from 1 to n, represents a node in the network with an unknown head, and each consists of an environment and learning automata where hydraulic analysis is executed using LA logic.

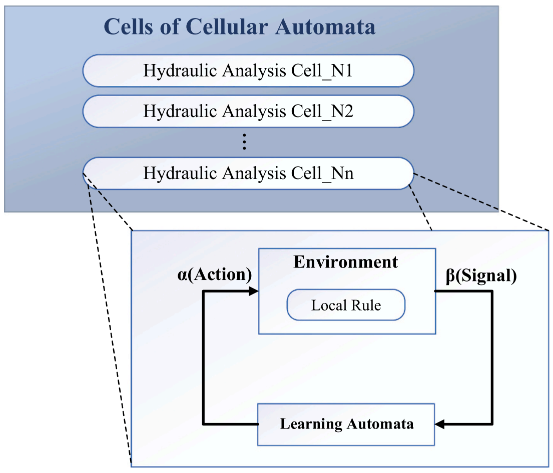

Cells of Cellular Automata
Hydraulic Analysis Cell_N1
Hydraulic Analysis Cell_N2
Hydraulic Analysis Cell_Nn
Environment
Local Rule
α(Action)
β(Signal)
Learning Automata

Fig. 5. LA-based steady-state hydraulic analysis of WDN.

3.5.1. *Convergence analysis*

In this section, we explore the convergence characteristics of our method, where the WDN is segmented into various sub-networks. The head of the central cell in each subnet is determined using Eq. (19). It's observed that zero flow in any pipe during steady state can lead to instability, as noted in [34]. However, our method, LA in CA, demonstrates robust convergence even in the presence of zero-flow conditions within the network. This resilience is attributed to our CA's update mechanism, where the head value calculations for a central cell incorporate the previous head values from adjacent cells at either end of the pipe. Consequently, the flow in such pipes does not reach zero until the model fully converges to a stable state.

We validated the convergence of our algorithm through two distinct approaches. Initially, we conducted simulations on a network consisting

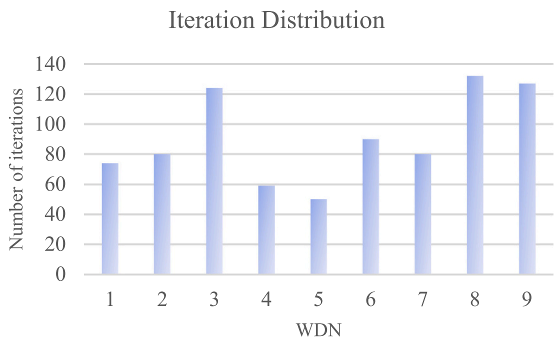


Fig. 6. Distribution of iterations required to reach convergence for different WDNs.

of five nodes and seven pipes, aligning our methods with the convergence analyses presented in [8], which examined various algorithms including GGA, LT, H (the Newton-Raphson nodal algorithm), and L (the simultaneous loop algorithm). The results of these simulations are depicted in Fig. 6, demonstrating that the number of iterations required to reach convergence varied from 50 to 132. This variation underscores the adaptability and effectiveness of our updating technique across different network scenarios.

Additionally, we employed the energy balance equation, a robust method to ensure the accuracy and stability of our simulations. This approach involves verifying the consistency of total power within the system between iterations, ensuring that the sum of internally dissipated power and the power delivered to users aligns with the total input power. This dual validation strategy not only reinforces the reliability of our algorithm but also showcases its capability to handle diverse and dynamic hydraulic conditions effectively within water distribution networks.

We rigorously apply the energy balance equation derived from Todini's research [35] to assess the convergence of our numerical model in hydraulic simulations. This method is a pivotal tool in verifying the efficiency and accuracy of the simulations within WDNs.

Convergence Analysis Using Energy Balance Equation:

The energy balance equation based on Todini's paper [35] is represented as $P_{tot} = P_{\text{int}} + P_{ext}$. This equation helps us understand how the total power input into the system (from sources like reservoirs) is distributed within the network. P_{int} represents the internal power dissipated within the pipes due to friction and other resistances, while P_{ext} reflects the power used effectively to deliver water to the end-users (nodes).

To ensure that our model provides reliable and stable solutions, we verify that the total energy across the network remains consistent between iterations. This involves checking the sum of P_{int} and P_{ext} against P_{tot} for all pipes and nodes in the WDN to ensure they balance within a small tolerance ε. A decreasing trend or stabilization in the difference between consecutive iterations of P_{tot} indicates that our model is approaching a physically realistic solution, demonstrating convergence and modeling accuracy. Eqs. (30) and (31) show the formulation for P_{int} for each pipe and P_{ext} for each node in the WDN, respectively.

$$P_{\text{int}} = H_f \tag{30}$$

$$P_{ext,node} = H_{d,node} * (q_{node}) \tag{31}$$

In these equations, H_f is each pipe's head-loss that is calculated using Eq. (1), H_d is the hydraulic head at each node and q_{node} is the node demand.

In order to check the quality of convergence, it is needed to sum up the energy losses due to head losses across all pipes ($\sum_{all\ pipes} \boldsymbol{P_{int}}$) and sum up the total power delivered to all nodes ($\sum_{all\ nodes} \boldsymbol{P_{ext}}$) and check its equality with total available power at the entrance in the WDN, which can be calculated using Eq. (32).

$$P_{tot} = \sum_{i=1}^{n_i} H_i Q_i \tag{32}$$

In this equation, Q_i and H_i are the flow rate and the head, respectively, for each reservoir i.

In each iteration, this equality suggests that the total energy inputs are consistent between iterations, indicating model stability. It shows that the numerical solutions are converging correctly, and the model is reliable for simulating the physical system. Utilizing the previously described formulations, we conducted extensive experiments to ensure that our convergence criteria were met for every WDN. This is particularly important as it highlights the effectiveness of our method in quickly achieving steady-state conditions, thereby validating the reliability and robustness of our computational approach in handling the complex dynamics of water distribution networks. Fig. 7 shows the convergence analysis using energy balance equation for Todini WDN [36].

The figure illustrates the progression of convergence in our simulation by plotting the difference between P_{tot} and ($P_{\text{int}} + P_{ext}$) over successive iterations. Each iteration shows a reduction in this difference, demonstrating that the system's energy balance is progressively improving. By the final iterations, the plot stabilizes near zero, indicating that the model has reached a steady state with well-balanced energy dynamics.

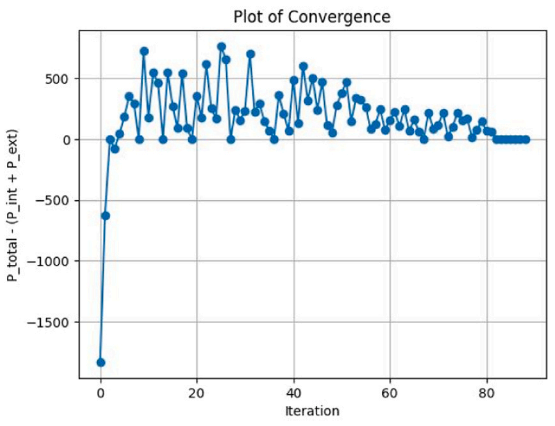


Fig. 7. The convergence of our method using energy balance equation for Todini WDN.

4. Numerical example

The previous section provided a thorough investigation of cellular and learning automata as well as hydraulic equations. The first row of Table 2 contains a figure illustrating a water distribution network, called Todini network which was selected from [36]. The proposed method is used to solve the hydraulic equations. This network comprises four demand nodes, seven pipes, and one source node. The resistance coefficient K of a pipe can be calculated using its diameter, length, and stiffness coefficient, as shown in Eq. (1). Table 1 provides the values of K and the nodal demands.

Based on the proposed LA-based method and Neumann's neighborhood structure in cellular automata, each node can have a minimum of one and a maximum of four neighbors. Consequently, the number of equations matches the number of non-source nodes. Table 2 provides a schematic representation of the Todini network, illustrating the process of forming parallel cells and their resolution using the Taylor series. The numbers with blue color are selected action in each probability vector in each iteration.

According to the proposed algorithm, each non-reservoir node and its neighbors are considered a cell; as a result, this WDN has four cells. All cells are simultaneously solved in each step based on the neighbor's current state. Then, the head values of the central nodes in all cells are updated. This process is repeated until head convergence is obtained for all nodes.

5. Results and discussion

Table 3 shows the calculated heads for the EPANET, CA_Linear_RTL [17], and LA in CA. The results of EPANET and LA in CA were obtained on a system with a 2.4 GHz processor and 4 gigabytes of memory, while the results of [17] were obtained on FPGA chip. The first column shows the node's number in the "Todini" network. The second and the third columns show the final values of the heads calculated by EPANET and CA_Linear_RTL architecture in [17], along with the calculation time for each. The fourth column shows the final head values calculated by the proposed method in this article, as well as the calculation time involved. The two last columns also show the accuracy of calculations in the

Table 1
Characteristics of junctions and pipes for the example.

Nodes	Head (m)	Demand (L/S)	Pipes	Length (m)	Roughness	Diameter (mm)	K
1	100	−100	1	1000	100	500	61.6851
2	1	10	2	1000	100	500	61.6851
3	1	20	3	1000	100	500	61.6851
4	1	30	4	1000	100	500	61.6851
5	1	40	5	1000	100	500	61.6851
-	-	-	6	1000	100	500	61.6851
-	-	-	7	1000	100	500	61.6851

proposed algorithm (head and flow balance errors) and the speed-up of calculation time compared to EPANET. The mass conservation (flow balance error) is calculated for each node using Eq. 2, based on the head value from the last iteration. Additionally, flow rates at each node are measured alongside the heads, providing a comprehensive view of the network's hydraulic performance. These flow measurements are crucial for confirming the functionality of the network under simulated conditions. Table 4 shows the calculated flows for the EPANET, [17], and LA in CA and energy conservation values for each pipe (Eq. 3). Moreover, Table 5 demonstrates the loops in the WDN and the energy conservation in loops (Eq. 4).

The energy conservation across each loop was evaluated using Eq. 4, which calculates the sum of head losses around each loop to ensure they approximate zero. This methodology confirms that the network adheres to the fundamental principles of hydraulic continuity and energy conservation, crucial for the stability and accuracy of hydraulic models in water distribution systems. Additionally, the total energy balance within the water distribution network (WDN) was rigorously analyzed and is detailed in Section 3.5.1, with results depicted in Fig. 7. This analysis verifies that the overall energy within the system is conserved across multiple simulation iterations, further substantiating the model's accuracy and stability in replicating realistic hydraulic conditions.

As part of the study, steady-state hydraulic analysis was applied to multiple WDNs from the University of Exeter site benchmarks [36]. In addition to evaluating the runtime and the number of iterations required to deliver results for three different algorithms, we also meticulously obtained the energy consumption associated with both our algorithm and EPANET. This comprehensive analysis allowed us to not only assess the computational efficiency but also evaluate the energy efficiency, highlighting the sustainable computing advantages of our method. By quantifying energy usage alongside runtime and iteration count, we offer a more holistic view of the performance benefits, further emphasizing the potential for significant energy savings in water distribution network analysis.

The Foss_Poly network, used as a key case study in our analysis, is illustrated in Fig. 8. This network representation helps visualize the complex structure and flow dynamics discussed.

This diagram of the Foss_Poly network shows all 37 nodes, including one reservoir (represented by node 37 as a red square), and the 58 pipes that interconnect these nodes. The nodes are labeled with identifiers, and arrows indicate the flow direction within the network. Notably, this network, typical of many WDNs, lacks pumps, tanks, and valves, which often simplifies the hydraulic analysis but also presents unique challenges in managing flow and pressure through design and operational strategies alone. This visual aid is fundamental for understanding the specific network dynamics and optimizing the discussed hydraulic models.

Since the main evaluation criteria of this research are the execution time and energy consumption, we have presented the execution time of EPANET, the computational architecture of CA_Linear_RTL presented in [17], and the architecture proposed in this article in one table and the energy consumption of EPANET and our algorithm in another table. Considering that the execution platform in the article [17] is an FPGA chip, with a different architecture and clock frequency, its runtimes and energy consumption may not be comparable with that of a general-purpose computing system. Therefore, we have compared the execution time, energy consumption, and acceleration rate of our approach with those of the EPANET on the same workstation.

Moreover, to provide a comprehensive evaluation, we have executed our proposed algorithm in both serial and parallel modes. This approach allows us to not only demonstrate the inherent efficiency of our algorithm's design but also showcases its scalability through parallel computation. The seventh column of Table 6 presents the runtime results for the serial implementation of our algorithm, enabling a direct comparison with the parallel execution times in the eighth column, thus highlighting the performance enhancements achievable through parallelization.

On the other hand, since the algorithm proposed in this article and the article [17] are both iterative, we have compared the proposed algorithm with the article [17] in terms of the number of iterations. The results show that by using the logic of LA in each cell of CA, we have been able to greatly reduce the number of iterations, which leads to reducing runtime and energy consumption. The first four columns of Table 6 show the characteristics of WDNs. Additionally, the combined runtimes of EPANET, CA_Linear_RTL, and our methods are presented under the *Runtime* category. The nineth column shows the maximum speed-up obtained compared to EPANET.

The proposed methods were evaluated for efficiency and accuracy using standard performance metrics. The results indicate that our (LA)-based methods accelerate the hydraulic analysis up to 60 times when used in the CA algorithm. Thanks to the parallel implementation of the cells, the proposed methods have significantly reduced execution time. As the accuracy evaluation reveals, the proposed methods yield results nearly identical to those obtained by EPANET, with our algorithm showing an error rate of 0.01 for head values.

Serial Implementation:

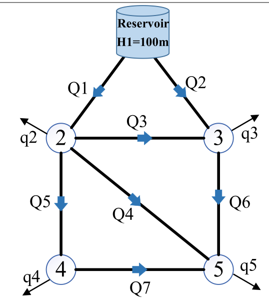
For the serial execution of our algorithm, we confined the computation to a single core. This was achieved by setting the concurrency settings in our computational environment to restrict the processing to only one processor. This method served as a baseline to assess the inherent efficiency of the algorithm's design without the advantages of multi-threading or parallel processing. The single-core execution allowed us to directly measure the performance impact of the algorithm's logic and computational structure in a constrained environment.

Parallel Implementation:

In contrast, the parallel implementation was facilitated through the use of a thread pool in Python, which effectively manages multiple threads across available CPU cores. This setup allows the operating system to allocate tasks dynamically to different processors, thus maximizing computational efficiency. By distributing tasks across multiple cores, the algorithm can perform multiple calculations concurrently, significantly reducing the overall execution time compared to the serial approach.

The thread pool approach is particularly advantageous for our CA based algorithm, where each cell of the CA operates independently. This independence is crucial for leveraging parallel processing, as it allows each cell to execute without waiting for other cells to complete their tasks. Consequently, this reduces idle time and enhances resource utilization, showcasing the scalability of our approach in handling larger and more complex network simulations.

Table 2
Todini WDN and concurrent sets of equations to be solved.



Demands	$q(H1) = 0$	$q(H2) = 0.01$	$q(H3) = 0.02$	$q(H4) = 0.03$	$q(H5) = 0.04$
Heads	$H1 = 100$	$(H2)^{(0)} = 3$	$(H3)^{(0)} = 4$	$(H4)^{(0)} = 5$	$(H5)^{(0)} = 6$

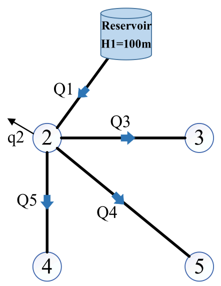
Probability Vectors (iteration 0):
N2 = [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]
N3 = [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]
N4 = [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]
N5 = [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]

$$\boldsymbol{f((H2)^0) = sign(|\frac{H1-H2}{k1}|^{\alpha} + |\frac{H3-H2}{k3}|^{\alpha} + |\frac{H4-H2}{k5}|^{\alpha} + |\frac{H5-H2}{k4}|^{\alpha}) - [q(H2)]}$$

$$\boldsymbol{f((H3)^0) = sign(|\frac{H1-H3}{k2}|^{\alpha} + |\frac{H2-H3}{k3}|^{\alpha} + |\frac{H5-H3}{k6}|^{\alpha}) - [q(H3)]}$$

$$\boldsymbol{f((H4)^0) = sign(|\frac{H2-H4}{k5}|^{\alpha} + |\frac{H5-H4}{k7}|^{\alpha}) - [q(H4)]}$$

$$\boldsymbol{f((H5)^0) = sign(|\frac{H2-H5}{k4}|^{\alpha} + |\frac{H4-H5}{k7}|^{\alpha} + |\frac{H3-H5}{k6}|^{\alpha}) - [q(H5)]}$$



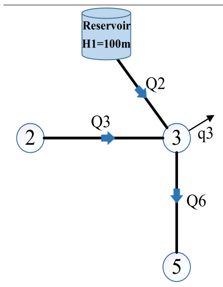
N2 (iteration 0) = [0.1, 0.1, 0.1, 0.1, **0.1**, 0.1, 0.1, 0.1, 0.1, 0.1]
N2 (iteration 1) = [0.05, 0.05, 0.05, 0.05, **0.55**, 0.05, 0.05, 0.05, 0.05, 0.05]

$$f = sign(|\frac{100-3}{61.6851}|^{0.54} + |\frac{4-3}{61.6851}|^{0.54} + |\frac{5-3}{61.6851}|^{0.54} + |\frac{6-3}{61.6851}|^{0.54}) - 0.01 = 1.66$$

$$df = (-0.54)(|\frac{100-3}{61.6851}|^{-0.46} + |\frac{4-3}{61.6851}|^{-0.46} + |\frac{5-3}{61.6851}|^{-0.46} + |\frac{6-3}{61.6851}|^{-0.46}) = -10$$

$$(H2)^{(1)} = 3 - \mathrm{w} * \left(\frac{1.72}{-10}\right)$$

$$deltaH(H2) = (H2)^{(1)} - (H2)^{(0)}$$



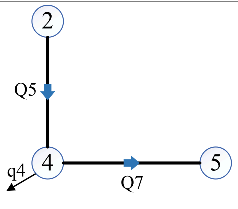
N3 (iteration 0) = [0.1, 0.1, 0.1, **0.1**, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]
N3 (iteration 1) = [0.1012, 0.1012, 0.1012, **0.0901**, 0.1012, 0.1012, 0.1012, 0.1012, 0.1012, 0.1012]

$$f = sign(|\frac{100-4}{61.6851}|^{0.54} + |\frac{3-4}{61.6851}|^{0.54} + |\frac{6-4}{61.6851}|^{0.54}) - 0.02 = 1.31$$

$$df = (-0.54)(|\frac{100-4}{61.6851}|^{0.54-1} + |\frac{3-4}{61.6851}|^{0.54-1} + |\frac{6-4}{61.6851}|^{0.54-1}) = -3.49$$

$$(H3)^{(1)} = 4 - \boldsymbol{w} * \left(\frac{1.31}{-3.49}\right)$$

$$deltaH(H3) = (H3)^{(1)} - (H3)^{(0)}$$



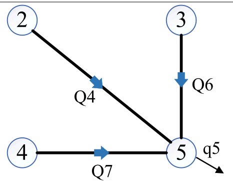
N4 (iteration 0) = [0.1, 0.1, 0.1, 0.1, **0.1**, 0.1, 0.1, 0.1, 0.1, 0.1]
N4 (iteration 1) = [0.1012, 0.1012, 0.1012, 0.1012, **0.0901**, 0.1012, 0.1012, 0.1012, 0.1012, 0.1012]

$$f(H4) = sign(|\frac{3-5}{61.6851}|^{0.54} + |\frac{6-5}{61.6851}|^{0.54}) - 0.03 = -0.1$$

$$df(H4) = (-0.54)(|\frac{3-5}{61.6851}|^{0.54-1} + |\frac{6-5}{61.6851}|^{0.54-1}) = -7.18$$

$$(H4)^{(1)} = 5 - \boldsymbol{w} * \left(\frac{-0.1}{-7.18}\right)$$

$$deltaH(H4) = (H4)^{(1)} - (H4)^{(0)}$$



N4 (iteration 0) = [0.1, 0.1, 0.1, 0.1, 0.1, **0.1**, 0.1, 0.1, 0.1, 0.1]
N4 (iteration 1) = [0.1012, 0.1012, 0.1012, 0.1012, 0.1012, **0.0901**, 0.1012, 0.1012, 0.1012, 0.1012]

$$f(H5) = sign(|\frac{3-6}{61.6851}|^{0.54} + |\frac{5-6}{61.6851}|^{0.54} + |\frac{4-6}{61.6851}|^{0.54}) - 0.04 = -0.48$$

$$df(H5) = (-0.54)(|\frac{3-6}{61.6851}|^{0.54-1} + |\frac{5-6}{61.6851}|^{0.54-1} + |\frac{4-6}{61.6851}|^{0.54-1}) = -8.37$$

$$(H5)^{(1)} = 6 - \boldsymbol{w} * \left(\frac{-0.48}{-8.37}\right)$$

$$deltaH(H5) = (H5)^{(1)} - (H5)^{(0)}$$

$$\boldsymbol{Convergence\ Condition: At\ least\ one\ entry\ in\ each\ probablity\ vector > 0.9}$$

Probability Vectors (iteration *n*):

[**0.9394**, 3.408e-13, 3.407e-13, 3.409e-13, 3.406e-13, 3.405e-13, 3.412e-13, 3.407e-13, 3.405e-13, 3.407e-13]
[**0.9986**, 1.585e-16, 1.576e-16, 1.574e-16, 1.578e-16, 1.576e-16, 1.541e-16, 1.573e-16, 1.578e-16, 1.578e-16]
[4.998e-09, 4.949e-09, 4.942e-09, 6.408e-09, **0.99764**, 4.859e-09, 6.682e-07, 4.945e-09, 6.074e-09, 4.942e-09]
[**0.9782**, 3.895e-13, 3.895e-13, 1.533e-12, 3.894e-13, 3.896e-13, 1.862e-12, 3.894e-13, 3.896e-13, 3.833e-13]

Table 3
Todini Network Head Calculation Results.

	EPANET	CA_Linear_RTL [17]	LA in CA	Difference (EPANET – LA in CA)	Mass Conservation
#	Head (m)	Head (m)	Head (m)	ΔH	Flow Balance Error
1	100	100	100	0	0
2	99.757	99.752	99.748	0.009	0.00022
3	99.762	99.759	99.753	0.009	0.00016
4	99.696	99.676	99.685	0.011	0.00013
5	99.701	99.698	99.689	0.012	0.00019
	Runtime (ms)	Runtime (ms)	Runtime (ms)	Speed-up	Average Accuracy
	114.5051	0.145	3.81	30	0.0001

Table 4
Todini Network Flow Calculation Results.

	EPANET	CA_Linear_RTL [17]	LA in CA	Difference (EPANET – LA in CA)	Energy Conservation
#	Flow (L/S)	Flow (L/S)	Flow (L/S)	ΔQ	Energy Balance Error
1	50.271	50.272	50.247	0.024	0.252
2	49.728	49.729	49.700	0.028	0.247
3	−6.067	−6.067	−6.051	0.016	0.005
4	22.613	22.612	22.621	0.008	0.063
5	23.726	23.727	23.727	0.001	0.064
6	23.660	23.660	23.698	0.038	0.004
7	−6.273	−6.274	−6.276	0.003	0.068

Table 5
Todini Network Loop Energy Conservation.

Loop	Path (Node Number)	Head loss	Loop Energy Conservation
1	2 → 1 → 3 → 2	0.252 − 0.247 + 0.005	0.010
2	3 → 5 → 2 → 3	0.064 −0.059 + 0.005	0.010
3	3 → 1 → 4 → 3	−0.247 + 0.315 −0.068	0.000

Performance Metrics:

The execution times recorded for both serial and parallel implementations, as presented in Table 6, illustrate the substantial speed improvements achieved through parallel processing. The comparative analysis not only highlights the effectiveness of employing multi-core environments but also underscores the potential for our CA-based algorithm in scalable, high-performance computing scenarios.

In this paper, we conducted thorough analysis of energy consumption for our algorithm, labeled "LA in CA", and performed a comparative analysis with the EPANET software, a standard in the field, across a variety of water distribution networks (WDNs). Our evaluation of energy consumption encompassed both package energy and DRAM energy consumption, providing a detailed assessment during the hydraulic analysis of several benchmark WDNs for both algorithms. This methodical evaluation allowed us to capture a comprehensive understanding of the energy efficiency inherent in each algorithm. The first column of Table 7 shows the names of the WDNs. The second and third columns present the energy consumption for hydraulic analysis using EPANET and LA in CA, respectively. The results indicate that incorporating LA in each CA cell can significantly reduce energy consumption.

As demonstrated in Table 7, the energy consumption metrics for various network sizes clearly indicate the advantages of our LA in CA approach, especially in larger and more complex network scenarios. This data underscores the significant energy efficiency improvements achieved by our method compared to traditional approaches like EPANET and serial implementation of ours. For instance, while smaller networks exhibit modest energy savings, the results from larger networks such as Jilin, BLA, and Foss_Poly show that our method can achieve substantial reductions in energy consumption. These findings are particularly relevant as they highlight the scalability of our approach, where the benefits become increasingly pronounced as the network size and complexity grow. This scalability is critical for urban and regional water distribution systems where energy efficiency can translate into significant cost and resource savings over time. Our approach not only reduces the computational time but also enhances energy efficiency without compromising the accuracy or the effectiveness of the hydraulic analysis.

By integrating LA in each CA cell, our method reduces the number of iterations required for analysis, further contributing to energy savings. This underscores the potential of our method to contribute to more

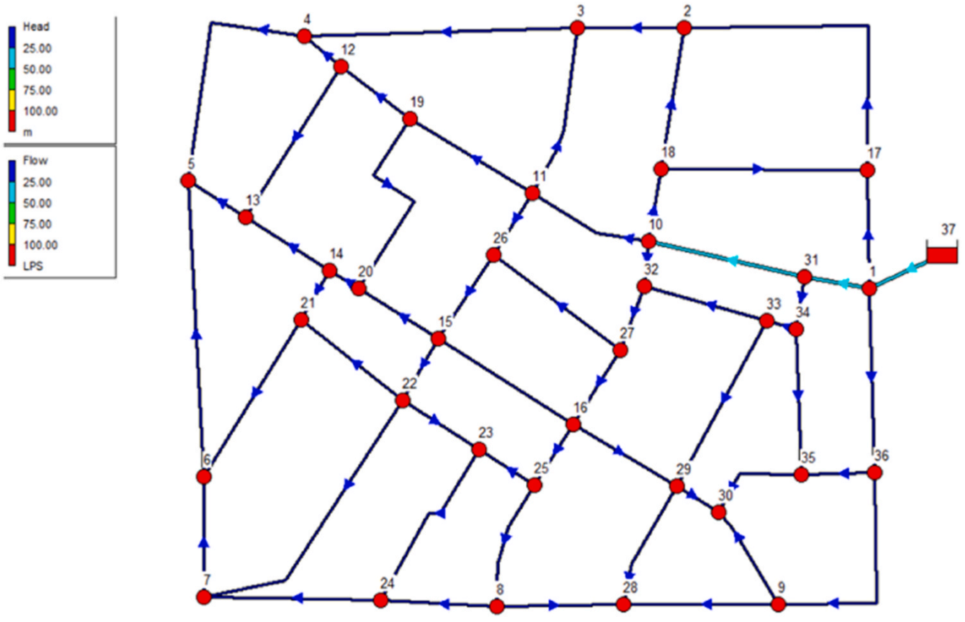


Fig. 8. Schematic Diagram of the Foss_Poly WDN.

Table 6
Benchmark Characteristics, Runtime achieved by the EPANET, CA_Linear_RTL, and our approach.

				Runtime (ms)				
						LA in CA		
WDNs	# of nodes	# of pipes	# of reservoirs	EPANET	CA_Linear_RTL [17]	Serial	Parallel	Speed-up
Noflow	4	3	2	110.56	0.241	12.33	2.77	39
Todini	5	7	1	114.50	0.145	13.07	3.81	30
Cheung	5	4	1	113.16	0.487	19.91	5.14	22
Example	5	5	1	116.53	0.350	13.09	1.94	60
Schur	6	7	1	115.11	0.174	12.75	3.87	29
Elhay_Zeroflow	8	11	1	113.04	0.250	13.67	4.96	22
Jilin	28	34	1	124.10	1.230	43.49	20.32	6
BLA	30	35	1	124.43	0.980	38.25	29.10	4
Foss_Poly	37	58	1	128.04	0.507	90.23	25.59	5
MOD	272	317	4	154.43	2.05	141.42	51.48	3

Table 7
The Energy Consumption achieved by EPANET and our approach.

	Energy (J)	
WDNs	EPANET	LA in CA
Noflow	0.83	0.21
Todini	1.29	0.59
Cheung	17.63	0.41
Example	15.65	0.26
Schur	6.24	0.33
Elhay_Zeroflow	26.41	0.59
Jilin	146.56	2.30
BLA	179.68	1.99
Foss_Poly	240.81	4.34
MOD	389.42	7.89

Table 8
The number of iterations achieved by the CA_Linear_RTL and our approach.

	No. of Iterations	
WDNs	CA _Linear_RTL [17]	LA in CA
Noflow	3869	74
Todini	2524	83
Cheung	2638	124
Example	2746	59
Schur	2520	50
Elhay_Zeroflow	5486	90
Jilin	5811	80
BLA	6124	132
Foss_Poly	6032	127
MOD	13587	261

sustainable water distribution network analysis practices by optimizing energy consumption without compromising on the accuracy or efficiency of the analysis. Based on this, Table 8 provides a detailed comparative analysis of the iterations required for hydraulic analysis using the CA_Linear_RTL method versus our LA in CA method. As detailed in the table, the application of LA in each CA cell leads to a notable reduction in the number of iterations needed to reach convergence, showcasing the effectiveness of our method in optimizing the computational workflow. The first column of Table 8 shows the names of the WDNs. The second and third columns present the number of iterations for hydraulic analysis using CA_Linear_RTL and LA in CA, respectively. The results indicate that incorporating LA in each CA cell can significantly reduce the number of iterations.

5.1. Discussion

Our parallel computational method with learning automata is highly applicable to various real-world scenarios, particularly in urban water management and emergency response planning. It offers computational efficiency and energy savings, and assists in optimizing network performance. However, the current demand-driven model may not accurately simulate conditions in networks where pressure variations are significant, such as in high elevation areas. Future developments will focus on integrating a pressure-driven analysis to improve the model's robustness and address non-uniform demand distributions. These enhancements will expand the model's applicability to more complex scenarios, including EPS and incorporating dynamic components in the model as detailed in previous sections.

5.1.1. Validation of conservation principles in network modeling

In our research, we have meticulously adhered to both mass and energy conservation principles within the framework of CA and LA. Our methodology leverages the iterative capabilities of CA, enhanced by the adaptive mechanisms of LA, to refine the solution accuracy over successive iterations.

Mass Conservation: Our model ensures mass conservation (Eq. 2) by validating that the algebraic sum of inflows and outflows at each node balances to zero, accounting for any external inflows or demands. This is achieved through iterative recalculations within each CA cell, adjusting the flow rates until the mass conservation equation is satisfied.

Energy Conservation: Our methodology rigorously adheres to energy conservation principles across the network. This is demonstrated through Section 3.5.1, where we detail the iterative process leading to energy balance across the network. Fig. 7 illustrates the convergence of energy values to zero, showcasing the effectiveness of our method in maintaining energy conservation within the Todini network. The iterative updates facilitated by the LA help in fine-tuning these values to reflect accurate energy states. Furthermore, Table 4 and Table 5 show the energy conservation for pipes and loops in the WDN.

Iterative Convergence and Accuracy: Utilizing the CA's capability for simultaneous equation solving within network cells, combined with the LA's reward-punish mechanism, our model efficiently linearizes and solves the network equations. This approach not only accelerates the computational process but also maintains high accuracy levels comparable to traditional methods like EPANET.

Accuracy Validation: We have enhanced our model validation by including additional columns in Tables 3 and 4 that report the mass conservation metrics at each node and energy conservation at each pipe. Moreover, Table 5 demonstrates the energy conservation in the loops of the network. This allows us to quantitatively demonstrate the conservation accuracy within our simulations, further substantiating the reliability of our reconstructed equations.

6. Conclusion

The development and validation of an efficient parallel computing algorithm utilizing LA and CA propose a significant advancement in the sustainable hydraulic analysis of water distribution networks. By partitioning WDNs into a set of cells based on CA principles and solving them in parallel using LA alongside Taylor series linearization, this study has demonstrated substantial reductions in computational time and energy consumption. Our method achieves up to 60 times faster analysis and with lower energy consumption compared to traditional methods such as EPANET. Experimental results show that our method delivers faster results and reduces energy consumption across networks of varying complexity. While smaller networks show modest energy savings, larger and more complex networks exhibit significant reductions. This dual advantage of speed and energy efficiency enhances the sustainability of computing resources, enabling rapid, precise, and effective analysis of water distribution networks without sacrificing quality. This study has validated and refined the demand-driven analysis approach. Future research will extend this methodology to encompass pressure-driven analyses and explore implementation on FPGA chips for further enhancements in speed and efficiency. This approach underscores the potential of sustainable computing to improve the environmental and societal impact of computational water resource management.

CRediT authorship contribution statement

Saheb Zamani Morteza: Writing – review & editing, Writing – original draft, Validation, Supervision, Resources, Project administration, Methodology, Formal analysis, Conceptualization. **Ahmadi Ruhollah:** Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Investigation, Formal analysis, Data curation. **Meybodi Mohammad Reza:** Writing – original draft, Validation, Supervision, Resources, Project administration, Methodology, Formal analysis, Conceptualization. **Suvizi Ali:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] G. He, T. Zhang, F. Zheng, Q. Zhang, An efficient multi-objective optimization method for water quality sensor placement within water distribution systems considering contamination probability variations, Water Res. 143 (2018) 165–175, https://doi.org/10.1016/j.watres.2018.06.041.

[2] H. Liu, D.A. Savić, Z. Kapelan, E. Creaco, Y. Yuan, Reliability surrogate measures for water distribution system design: comparative analysis, J. Water Resour. Plan. Manag. 143 (2) (2017) 04016072, https://doi.org/10.1061/(ASCE)WR.1943-5452.0000728.

[3] H. Yan, Q. Wang, J. Wang, K. Xin, T. Tao, S. Li, A simple but robust convergence trajectory controlled method for pressure driven analysis in the water distribution system, Sci. Total Environ. 659 (Apr. 2019) 983–994, https://doi.org/10.1016/j.scitotenv.2018.12.374.

[4] E. Todini, L.A. Rossman, Unified framework for deriving simultaneous equation algorithms for water distribution networks, J. Hydraul. Eng. 139 (5) (2013) 511–526, https://doi.org/10.1061/(ASCE)HY.1943-7900.0000703.

[5] H. Cross, Analysis of Flow in Networks of Conduits or Conductors, Bulletin, University of Illinois at Urbana Champaign, College of Engineering, Illinois, 1936. Available: ⟨http://hdl.handle.net/2142/4433⟩.

[6] D. Martin, G. Peters, The application of Newton's method to network analysis by digital computer, J. Inst. Water Eng. 17 (2) (1963) 115–129.

[7] D.J. Wood and J.E. Funk, Hydraulic Analysis of Water Distribution Systems, Proceedings International Conference on Water Supply Systems, State of the Art and Future Trends, Valencia, Spain, Oct. 1992.

[8] E. Todini, On the convergence properties of the different pipe network algorithms (Cincinnati, Ohio, United States, March), Water Distrib. Syst. Anal. Symp. (2008) 1–16, https://doi.org/10.1061/40941(247)75.

[9] L.A. Rossman, Epanet 2 users manual, in: Water Supply and Water Resources Division, 45268, National Risk Management Research Laboratory, Cincinnati, OH, 2000. ⟨https://www.epa.gov/sites/production/files/2015-09/documents/epanet2-users-manual-2_0.pdf⟩ ([Online]. Available).

[10] M. Guidolin, Z. Kapelan, D. Savic, Using high-performance techniques to accelerate demand-driven hydraulic solvers, J. Hydroinformatics 15 (1) (2013) 38–54, https://doi.org/10.2166/hydro.2012.198.

[11] J. Peinado and A.M. Vidal, Three Parallel Algorithms for Solving Non-Linear Systems and Optimization Problems, International Conference on High-Performance Computing for Computational Science, Springer, pp. 657-670, Berlin, Heidelberg, June. 2004, doi: 10.1007/11403937_49.

[12] A. Suvizi, S. Subramaniam, T. Lan and G. Venkataramani, Exploring In-Memory Accelerators and FPGAs for Latency-Sensitive DNN Inference on Edge Servers, in IEEE Cloud Summit, Washington, DC, USA, 2024.

[13] K. Diao, Z. Wang, G. Burger, C.-H. Chen, W. Rauch, Y. Zhou, Speed-up of water distribution simulation by domain decomposition, Environ. Model. Softw. 52 (2014) 253–263, https://doi.org/10.1016/j.envsoft.2013.09.025.

[14] A. Toselli, O. Widlund, Domain Decomposition Methods: Algorithms and Theory, Springer Science & Business Media, 2006. ⟨https://link.springer.com/book/10.1007/b137868⟩ ([Online]. Available).

[15] Z. Vasilic, D. Ivetić, Ž. Vasilić, M. Stanić, D. Prodanović, Speeding up the water distribution network design optimization using the ΔQ method, J. Hydroinformatics 18 (1) (2016) 33–48.

[16] Z. Vasilic, M. Stanic, Z. Kapelan, D. Ivetic, D. Prodanovic, Improved loop-flow method for hydraulic analysis of water distribution systems, J. Water Resour. Plan. Manag. 144 (4) (2018), https://doi.org/10.1061/(ASCE)WR.1943-5452.0000922.

[17] A. Suvizi, A. Farghadan, M. Saheb Zamani, A parallel computing architecture based on cellular automata for hydraulic analysis of water distribution networks, J. Parallel Distrib. Comput. 178 (2023) 11–28, https://doi.org/10.1016/j.jpdc.2023.03.009.

[18] A.W. Burks, Essays on Cellular Automata, University of Illinois Press, Urbana, 1970.

[19] U. Shamir, C.D. Howard, Engineering analysis of water-distribution systems, J. Am. Water Works Assoc. 69 (7) (1977) 510–514, https://doi.org/10.1002/j.1551-8833.1977.tb06802.x.

[20] T. Walski, D.V. Chase, D. Savic, W.M. Gray Man, S. Beckwith, E. Koeller, and Haestad Methods, Advanced Water Distribution Modeling and Management, 2003. [Online]. Available: ⟨https://core.ac.uk/download/pdf/232829091.pdf⟩.

[21] N. Moosavian, B.K. Roodsari, Soccer league competition algorithm: a new method for solving systems of non-linear equations, Int. J. Intell. Sci. 4 (1) (2014) 7–16, https://doi.org/10.4236/ijis.2014.41002.

[22] E. Todini, S. Pilati, A gradient algorithm for the analysis of pipe networks, Comput. Appl. Water Supply. Syst. Anal. Simul. 1 (1988) 1–20.

[23] O. Giustolisi, D. Laucelli, L. Berardi, D.A. Savic, A computationally efficient modeling method for large water network analysis, J. Hydraul. Eng. 138 (9) (2012) 1044–1053, https://doi.org/10.1061/(ASCE)HY.1943-7900.0000517.

[24] J.Von Neumann, A.W. Burks, Von Neumann's self-reproducing automata, Essays Cell. Autom. (1969) 4–65. ⟨https://fab.cba.mit.edu/classes/865.18/replication/Burks.pdf⟩.

[25] M. Afshar, M. Rohani, Optimal design of sewer networks using cellular automata-based hybrid methods: discrete and continuous approaches, Eng. Optim. 44 (1) (2012) 1–22, https://doi.org/10.1080/0305215X.2011.557071.

[26] E.M. Khaneghah, A.R. ShowkatAbad, N. Shadnoush, N. Ismayilova, R. N. Ghahroodi, E. Ismayilov, M.S.N. Saravani, F.T. Sarraf, A. Soveizi, ExaMig matrix: process migration based on matrix definition of selecting destination in distributed exascale environments, Azerbaijan J. High. Perform. Comput. 1 (1) (2018) 20–41.

[27] S. Wolfram, Cellular automata and complexity: collected papers, 152, Addison-Wesley, 1994.

[28] M.H. Afshar, A cellular automata approach for the hydro-power operation of multi-reservoir systems, Proc. Inst. Civ. Eng. -Water Manag. 166 (4) (2013) 465–478, https://doi.org/10.1680/wama.11.00105.

[29] K.S. Narendra, M.A.L. Thathachar, Learning Automata: An Introduction, Prentice-Hall, 1989.

[30] M.A.L. Thathachar, P.S. Sastry, Networks of Learning Automata: Techniques for Online Stochastic Optimization, Kluwer Academic Publishers, 2004.

[31] K.S. Narendra, M. Thathachar, Learning automata: a survey, IEEE Trans. Syst. Man Cyber (4) (1974) 323–334.

[32] A. Rezvanian, A.M. Saghiri, S.M. Vahidipour, M. Esnaashari, and M.R. Meybodi, Recent Advances in Learning Automata, vol. 754. in Studies in Computational Intelligence, vol. 754. Cham: Springer International Publishing, 2018. doi: 10.1007/978-3-319-72428-7.

[33] L. Musmade, M. Chidambaram, Learning automata based set-point weighted parameter for unstable systems, IFAC Proc. Vol. 47 (1) (Jan. 2014) 122–126, https://doi.org/10.3182/20140313-3-IN-3024.00162.

[34] N. Moosavian, Multilinear method for hydraulic analysis of pipe networks, J. Irrig. Drain. Eng. 143 (5) (May 2017) 04017020, https://doi.org/10.1061/(ASCE)IR.1943-4774.0001163.

[35] E. Todini, Looped water distribution networks design using a resilience index-based heuristic approach, Urban Water 2 (2) (Jun. 2000) 115–122, https://doi.org/10.1016/S1462-0758(00)00049-2.

[36] The University of Exeter, Benchmarks, [Online]. Available: ⟨https://engineering.exeter.ac.uk/research/cws/resources/benchmarks/⟩.