Engineered Science
DOI : https://dx.doi.org/10.30919/es1061

A Comprehensive Review of the de Bruijn Graph and Its Interdisciplinary Applications in Computing

Shien Huang,[1] Hang Zhang[2,*] and Ergude Bao[1,*]

Abstract

The de Bruijn graph was proposed by de Bruijn and Good in 1946. It was initially employed in binary sequence research but has been widely adopted across diverse fields and disciplines like information and coding theory, computational science, hardware architecture, and distributed networks. However, a comprehensive review of its various applications is lacking. To address this issue and to honor the 105th birth anniversary of Nicolaas Govert de Bruijn, we present this review. In this review, we categorize de Bruijn graph's features into node/edge features and structure features, and then describe its applications accordingly. Applications utilizing node/edge features include pseudo-random sequence design, correction code design for racetrack memory, image design for self-location in tangible user interface, reversibility determination for one-dimensional cellular automata, and genome assembly, while those utilizing structure features involve VLSI design, connection among wireless sensors, and network design with distributed hash tables. In addition, we also summarize variations of the de Bruijn graph for enhanced performance, including positional de Bruijn graph and A-Bruijn graph in genome assembly, constrained de Bruijn graph in correction code design for racetrack memory, and hierarchical de Bruijn graph in network design with distributed hash tables. At last, we propose some ideas of the de Bruijn graph's novel applications.

Keywords: De Bruijn graph; Information and coding theory; Computational science; Hardware architecture; Distributed networks.
Received: 09 November 2023; Revised: 11 December 2023; Accepted: 12 December 2023.
Article type: Review article.

1. Introduction

The de Bruijn graph is a directed graph where nodes represent fixed-length sequences of given symbols, and two nodes with overlapping sequences are connected through an edge. Origin of the de Bruijn graph can be traced back to de Bruijn and Good's work in 1946,[1,2] where it was initially proposed to investigate problems related to binary sequences, and has close relationship with the feedback shift registers (FSR; for more details about the FSR, please refer to SI Appendix, Section Relationship between the FSR and de Bruijn graph). With advancement of researches, the de Bruijn graph has been applied in various fields and disciplines like information and coding theory, computational science, hardware architecture, and distributed networks. However, there is currently a lack of comprehensive review of the de Bruijn graph's applications across different fields. Such a review is important because a comprehensive summary of existing applications could inspire novel applications of the de Bruijn graph. To address this issue and to honor Nicolaas Govert de Bruijn's 105th birth anniversary, we present this review. In this review, we initially categorize features of the de Bruijn graph into node/edge features and structure features. The former features mainly include each node representing a unique sequence and two connected nodes representing overlapping sequences. The latter features are such as the graph diameter is relatively small and number of out-neighbors for each node is constant. Then we classify the de Bruijn graph's applications accordingly based on the features. The applications utilizing node/edge features include pseudo-random sequence design,[3–9] correction code design for racetrack memory,[10–12] image design for self-location in tangible user interface,[13–19] reversibility determination for one-dimensional cellular automata,[20–31] and genome assembly;[32–64] the applications

[1] *School of Software Engineering, Beijing Jiaotong University, Beijing 100044, China.*
[2] *Institute of Engineering Thermophysics, Chinese Academy of Sciences, Beijing 100190, China.*
**Email:* zhanghang@iet.cn (H. Zhnag), baoe@bjtu.edu.cn (E. Bao)

utilizing structure features include VLSI design,[65–75] connection among wireless sensors,[76–83] and network design with distributed hash tables.[84–90] In addition, we also discuss variations of the de Bruijn graph that appear in these applications, including positional de Bruijn graph and A-Bruijn graph in genome assembly, constrained de Bruijn graph in correction code design for racetrack memory, and hierarchical de Bruijn graph in network design with distributed hash tables, and propose some ideas of the de Bruijn graph's novel applications.

2. Theory of the de Bruijn graph

As a directed graph, each node u of the de Bruijn graph represents a sequence of length k over q given symbols. That is, each node u is $(x_1, x_2, \ldots, x_k)$, and x_i $(1 \le i \le k)$ belongs to a symbol set $\mathbb{F}_q$ containing q different symbols. In addition, two nodes u and v in the graph are connected by a directed edge from u to v if and only if the suffix of u of length $k-1$ is equal to the prefix of v. That is, any two node $u = (x_1, x_2, \ldots, x_k)$ and node $v = (x_2, \ldots, x_k, x_{k+1})$ are connected with an edge in the graph. According to the above definition, a de Bruijn graph can be determined by the length k and symbol set $\mathbb{F}_q$, so we refer to such de Bruijn graph as k-dimensional de Bruijn graph with $\mathbb{F}_q$ or k-dimensional de Bruijn graph over q symbols. Specifically, if $\mathbb{F}_q = \{0, 1\}$, the de Bruijn graph is called binary de Bruijn graph.

According to the definition of k-dimensional de Bruijn graph over q symbols, we can summarize some node/edge features of the graph. Firstly, each node of the de Bruijn graph represents a unique sequence of length k, and there are q^k nodes representing all k-length sequences over symbol q. Secondly, sequences of two connected nodes have an overlap of length $k-1$, so each connecting edge represents a unique sequence of length $k+1$. Based on the node/edge features, a related concept de Bruijn sequence can be introduced. The de Bruijn sequence is a circular sequence obtained by traversing all nodes of a de Bruijn graph, and it has the property that all subsequences of length k are unique. When the de Bruijn graph is binary de Bruijn graph, the de Bruijn sequence is referred to as binary de Bruijn sequence. It is worthwhile to note that, such a graph traversal problem is usually defined as Hamiltonian cycle problem, which is NP-Hard, but fortunately for the de Bruijn graph, the problem can be alternatively defined as Eulerian cycle problem, which can be solved in polynomial time.[91]

In addition to the aforementioned node/edge features, the de Bruijn graph has some structure features. Firstly, number of out-neighbors for each node is fixed q in the de Bruijn graph. Secondly, diameter of the graph, length of the shortest path between the most distanced nodes, is fixed k. Since number of nodes in the graph is q^k, k is sufficiently small equal to $\log_q q^k$. Thirdly, the shortest path between two nodes can be obtained in $O(k)$ time by comparing sequences represented by source node and destination node using a prefix tree.[92] For example, to obtain the shortest path from node 0010 to 1001, with the former's suffix 10 equal to the latter's prefix, the shortest path is obtained directly as 0010 → 0100 → 1001. Note that this feature is not used in any of the reviewed applications, but we feel it needs to be mentioned for completeness. Fourthly, with limited nodes removed from the de Bruijn graph, its connectivity can be largely kept. For example, when $q > 2$ and the number of removed nodes is less than or equal to $q-2$, the remaining nodes will remain connected. In case of $q = 2$, removing one node can leave at least $2^k - k - 1$ nodes

Table 1. Features of the k-dimensional de Bruijn graph over q symbols, and various applications of the de Bruijn graph based on its features.

Category	Feature description	Field and discipline	Application
Node/edge features	Each node represents a unique sequence of length k There are q^k nodes representing all k-length sequences over symbol q Sequences of two connected nodes have an overlap of length $k-1$ Each connecting edge represents a unique sequence of length $k+1$	Information and coding theory	Pseudo-random sequence design
			Correction code design for racetrack memory
			Image design for self-location in tangible user interface
		Computational Science	Reversibility determination for one-dimensional cellular automata
			Genome assembly
Structure features	Number of out-neighbors for each node is fixed q Diameter of the graph is fixed k The shortest path between two nodes can be obtained in $O(k)$ time With limited nodes removed from the graph, its connectivity can be largely kept	Hardware architecture	VLSI design
		Distributed networks	Connection among wireless sensors
			Network design with distributed hash table

connected.[93] These introduced node/edge features and structure features above are listed in Table 1.

3. Applications based on node/edge features

3.1 Information and coding theory

3.1.1 Pseudo-random sequence design

Pseudo-random sequences are binary sequences of certain length proposed along with the invention of computer in 1946,[3] and have various applications. For example, in wireless communication systems, they can be used as spreading codes;[4,5] in cryptography, they can be employed for stream ciphers;[6] in neuroimaging, they are utilized as ideal stimuli.[7] There are several typical requirements to design quality pseudo-random sequences.[8] (i) The sequence length is relatively large. (ii) Any subsequence of a specific length within a sequence is unique. (iii) Number of 0s and 1s in a sequence is equal. (iv) Number of various sequences is sufficiently large.

To meet these requirements, the k-dimensional binary de Bruijn graph was used to generate binary de Bruijn sequences as the pseudo-random sequences, at the time the de Bruijn graph was initially proposed. The reasons are as below. (1) The k-dimensional binary de Bruijn graph has 2^k nodes, so length of the generated sequences is 2^k relatively large. (2) Each subsequence of length k within a binary de Bruijn sequence corresponds to a node in the binary de Bruijn graph, so it is unique. (3) Nodes of the binary de Bruijn graph represent all k-length binary sequences, so a generated sequence has equal number of 0s and 1s. (4) Different traversals of the binary de Bruijn graph can generate various binary de Bruijn sequences, and the number of generated sequences is $2^{2^{k-1}-k}$ which is sufficiently large.[1]

Some typical alternative pseudo-random sequences are m-sequences, gold sequences and Walsh-Hadamard sequences.[5] The m-sequences can satisfy requirements (i), (ii) and (iii), but not (iv), whereas the gold sequences and Walsh-Hadamard sequences show some improvements in (iv), but still not sufficient. SI Appendix, Table S1 provides a comparison about number of various m-sequences, gold sequences, Walsh-Hadamard sequences and binary de Bruijn sequences with fixed length requirement. Though the de Bruijn sequences have many advantages, their implementation is not straightforward relying on graph traversal, and therefore limits further utilization. A few researches on how to efficiently implement the de Bruijn sequences have been proposed recently.[8,9]

3.1.2 Correction code design for racetrack memory

Racetrack memory is a type of computer memory first proposed by IBM Almaden Research Center in 2008, which uses multiple read/write heads for fast memory accesses with low power consumption.[10–12] To read data, multiple heads are used upon a moving array of bits simultaneously. During this process, the array may move at unexpected speeds, leading to over-shift (skipping some bits) or under-shift (reading the same bits) errors for all the read heads. To detect such errors, the racetrack memory includes a correction code and a head reading it. Two consecutive readings can obtain two m-length sequences: an offset of exactly one bit between the two sequences indicates no error; an offset of greater than one bit indicates an over-shift error; no offset indicates an under-shift error. Obviously, the correction code needs to be specifically designed to enable fast determination of the offset.

The binary de Bruijn sequence could be used as such correction code, because any subsequence of a fixed length $k < m$ in the binary de Bruijn sequence is unique, and the subsequence in one m-length sequence can be used to

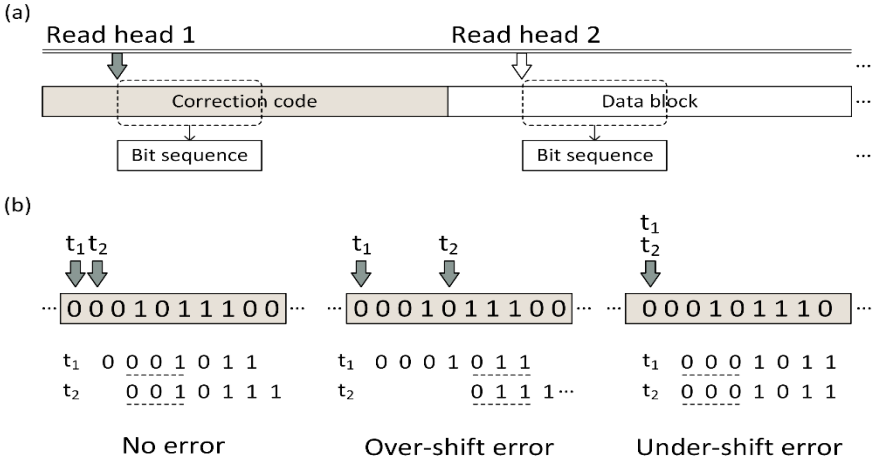

Fig. 1 (a) Illustration of the racetrack memory. (b) Examples of no error, over-shift error and under-shift error using binary de Bruijn sequence as correction code.

efficiently match the other m-length sequence. Fig. 1-(a) shows an illustration of the racetrack memory, and Fig. 1-(b) shows examples of no error, over-shift error and under-shift error using binary de Bruijn sequence as the correction code. In the examples, $m = 7$ and $k = 3$, and the two consecutive readings are indicated as t_1 and t_2, respectively. By matching first 3 bits of the 7-length sequence from reading t_2 with the 7-length sequence from reading t_1, offsets of one, four and zero can be determined in the examples, indicating no error, over-shift error (skipping 3 bits) and under-shift error, respectively. After an error is detected, all the read heads can be adjusted simultaneously to correct the error.

Alternative error correction methods are mainly based on the Varshamov-Tenengolts code.[94] Though they are simpler for implementation, the de Bruijn graph-based method can detect and correct over-shift errors of more than 2 skipped bits, and is thus more powerful. However, the racetrack memory is currently not an active research area, and the study of de Bruijn graph's application here has largely discontinued, but the existing results could be referenced for its applications in other areas.

3.1.3 Image design for self-location in tangible user interface

Tangible user interface contains tiny devices through which human and computer interact with each other.[13,14] The concept of tangible user interface was first proposed in 1997, and has been used in programming education, video games and music composition, etc.[15] The tiny devices usually need to locate themselves, and this can be achieved with sensors and specifically designed images.[16] Take the most simple tiny device computer mouse as an example, it can use an image placed on mouse pad to locate itself. To do this, it uses optical sensor to capture local portion of the image and then translates the local image to location with stored image-to-location information. Obviously, the image design is important for self-location. The designed image for self-location should not be complex, since the optical sensor in a tiny device capturing local images is usually simple. In addition, local portions of the designed image need to be unique for the image-to-location mapping.

For one-dimensional self-location, to design the image is essentially to design a sequence, and then convert each symbol in the sequence into an image block. There are some typical requirements to design the quality sequence. (i) The number of possible symbols at each position of the sequence should be minimized, with the ideal case only two symbols. (ii) Subsequences of a specific length in the designed sequence are unique. To meet these requirements, the binary de Bruijn sequence generated from a k-dimensional binary de Bruijn graph was used in the 1980s.[17] This is because the binary de Bruijn sequence (1) contains only 0 or 1 for each position, and (2) has unique subsequences of length k. Fig. 2 provides an example of using a de Bruijn sequence for such a designed image. In this example, the 0 and 1 in de Bruijn sequence can be converted to white and black blocks, respectively, and the sequence can be converted into an image.

An alternative method to design such a one-dimensional image is based on the Gray code.[95] Though this method has broader utilities such as locating rotating devices, symbols in the designed sequence are complex and does not meet requirement (i) above, so it is less frequently used in one-dimensional self-location. In the case of two-dimensional self-location, designing the image is essentially to design a matrix. Such a matrix can be designed by combining two binary de Bruijn sequences, where one is for horizontal dimension of the matrix and the other for the vertical dimension. For detailed information, please refer to the recent research papers.[16,18,19]

3.2 Computational science

3.2.1 Reversibility determination for one-dimensional cellular automata

Cellular automaton (CA) is a special mathematical model used in multiple disciplines.[20,21] It was proposed in the 1940s to study

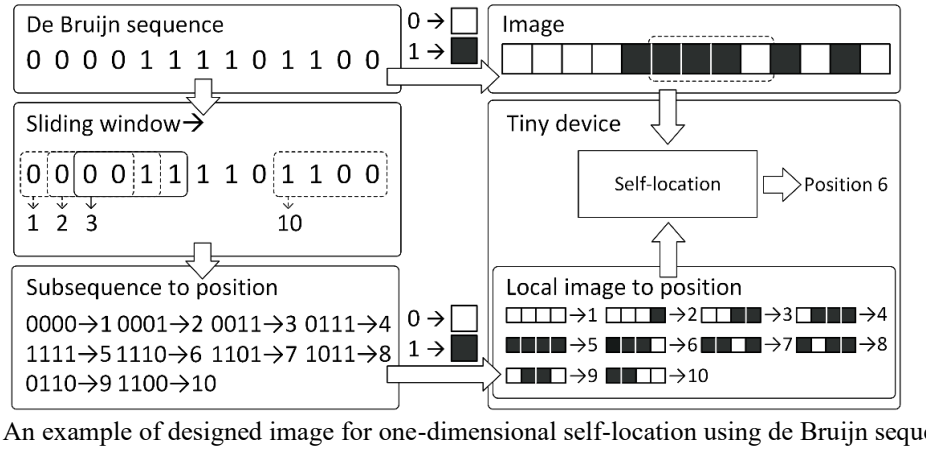

Fig. 2 An example of designed image for one-dimensional self-location using de Bruijn sequence.

the Turing machine, and began to gain widespread attention in 1970 in research of cell developments. Since that time, CA has been further applied in quite a few fields, such as geography and thermophysics. In geography, CA can be used to model land usage and forecast urban development.[22] In thermophysics, CA is recently utilized to model combustion processes or behavior of solar energy systems, aiming to optimize efficiency of energy utilization.[23,24] Among various versions of CA, one-dimensional reversible CA is a key focus,[25,26] which is capable of transforming a sequence of certain length into a new sequence of same length and can get back with another CA. Such CA has a *rule* to map one symbol together with its left r and right r symbols to a new symbol. For example, a rule called *rule 150* can map 000, 011, 101 and 110 to 0, and map 001, 010, 100 and 111 to 1.[27] To transform a sequence 011101 to 101001 with this rule, the process is as below. At first, the 011101 is padded with 0s in the beginning and end to form 00111010. Then, every three consecutive 0/1 from the 00111010 is mapped to either 0 or 1 according to the rule. Finally, the mapping results are concatenated to form the sequence 101001.

To determine the reversibility of one-dimensional CA, its rule needs to be represented by a directed graph with the following requirements.[28,29] (i) Nodes of the graph represent all sequences of length $2r$. (ii) Node u is connected with node v through an edge, if $2r-1$ suffix of u's represented sequence is the same with prefix of v's sequence. That is, each edge represents a sequence of length $2r+1$ merging the connecting nodes' sequences, and further represents mapping of the $2r+1$ symbols to a new symbol. Obviously, such a graph is a $2r$-dimensional de Bruijn graph. With the $2r$-dimensional de Bruijn graph representation for a CA's rule, existence of specific paths in the graph are studied to determine the CA's reversibility.[26,30,31] SI Appendix, Fig. S1 provides an example of the de Bruijn graph representing *rule 150*. It is worthwhile to mention that, the reversibility determination of one-dimensional CA is currently not an active research area, but the existing studies of de Bruijn graph's application could be referenced in other areas.

3.2.2 Genome assembly

Genome assembly is to assemble sequenced genome fragments into the complete genome for biology and medical researches.[32–34] To obtain a complete genome sequence, various sequencing technologies have emerged, but even the most up-to-date technology can only generate fragments of the genome named reads.[35–37] Fortunately, the number of sequencing reads is large, and many of them from the same or close genome positions share overlaps. Therefore, it is possible to assemble these reads to form a complete genome sequence, and methods of the genome assembly need to be studied.

To assemble the reads of a species into the genome, a relatively straight forward method is to construct a graph, where each node represents a read and each edge between two nodes represents two reads' overlap, and then traverse the graph to obtain the assembly result.[38–40] When the reads are long and the number is not very large, this method is feasible, but when the reads are short and the number is very large, computing overlaps between the reads is time-consuming and cannot be finished in acceptable time. Additionally, traversal of the constructed graph is to find a Hamiltonian cycle in it, and the Hamiltonian cycle problem is NP-hard. To address these challenges, the reads could be used to construct a k-dimensional de Bruijn graph instead.[41] The specific method is: firstly, each read is divided into consecutive subsequences of length k, called k-mers, using a sliding window; secondly, a k-dimensional de Bruijn graph is constructed with each node for all identical k-mers from different reads, and each edge for two consecutive k-mers from the same read; finally, the problem to traverse the de Bruijn graph is defined as Eulerian cycle problem to generate the assembled sequence.[42,43] This method is fast, because computing overlaps between reads is converted to locating identical k-mers from different reads, and the latter can be achieved in linear time with a hash table. Additionally, the Eulerian cycle problem can also be solved in polynomial time. In Fig. 3, workflow of the genome assembly based on the de Bruijn graph is illustrated. Note that it is usually not easy to directly assemble the reads into the complete genome, due to many reasons such as sequencing errors, gaps and repeat regions in the genome.

Various tools exist for genome assembly using de Bruijn graph. For relatively short reads, representative tools include Euler,[42] Velvet,[44] AbySS,[45] IDBA,[46] ALLPATHS-LG,[47,48] SPAdes,[49] SOAPdenovo2,[51,52] AlignGraph,[53] and ScalaDBG.[54] For relatively long reads, representative tools include wtdbg2,[55] Flye,[56] AlignGraph2,[57] MBG,[58] mdBG,[59] and LJA.[60]

4. Applications based on structure features

4.1 Hardware architecture

4.1.1 VLSI design

VLSI (Very Large-Scale Integrated circuit) is an integrated circuit comprising a significantly large number of transistors forming components such as processing elements.[65,66] The integrated circuit was initially invented by Texas Instruments in 1958, and its scale of integration has been improved from small to very large. With very large scale, the VLSI is powerful

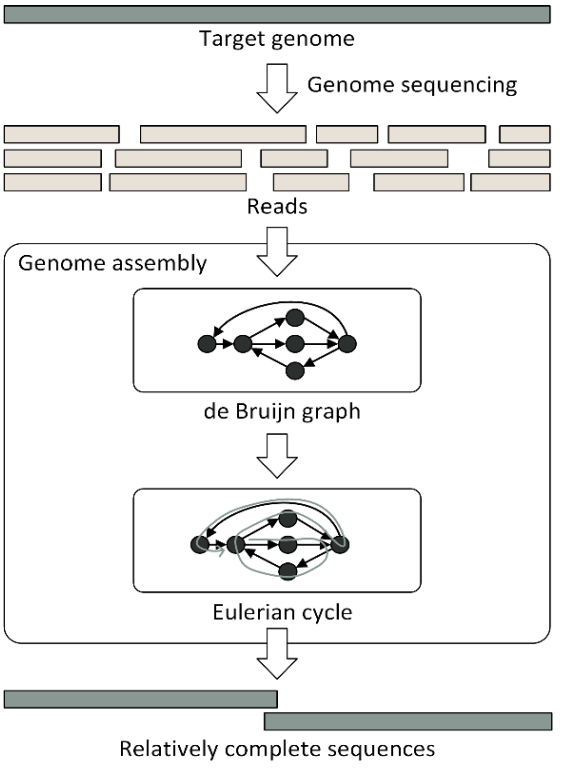

Fig. 3 Workflow of genome assembly based on de Bruijn graph.

in computation, and has been crucial in applications ranging from everyday smartphones to big engineering projects. To design a VLSI, there are several typical requirements for its components.[67] (i) Each component in the designed VLSI should be connected to a relatively small number of other components to reduce overall placement size of the VLSI. Ideally, each component has a constant number of connected components. (ii) The VLSI's latency and power consumption should be as low as possible. That is, the shortest distance between the most distanced components should be small. (iii) When a single electronic component fails in the VLSI, it should not result in serious disconnection of the remaining components or abnormal operations. That is, the components should be connected in a way to tolerate such failures.

To meet these requirements, the VLSI design was first modeled with the k-dimensional de Bruijn graph over q symbols in 1984, where each node of the graph represents a component, and each edge represents connection of two components.[68] The reasons are as below. (1) Number of out-neighbors for each node in the de Bruijn graph is fixed q, so the number of connected components in the VLSI is guaranteed constant. (2) Diameter of the de Bruijn graph is fixed k, indicating the shortest distance between the most distanced components in the VLSI is relatively small. (3) By removing a finite number of nodes, most remaining nodes in the de Bruijn graph are still connected, and this means a single component's failure in the VLSI does not affect connectivity of other components. SI Appendix, Fig. S2 shows an example of VLSI layout based on de Bruijn graph, and some design examples include sorting circuit design,[69] decoding circuit design[70,71] and network-on-chip design.[72,73]

The VLSI can be alternatively modeled as mesh, tree or hypercube structures.[96,97] These design methods are much simpler than the de Bruijn graph based method, but may not fully meet the requirements (i), (ii) and (iii). Specifically, in the mesh structure, the most distanced components are across the diagonal relatively far away, not meeting requirement (ii); in the tree structure, a single component is mostly connected with only two other components, so its failure will result in disconnection of the other components, not meeting requirement (iii); in the hypercube structure, though each component has fixed number of connected components, the number varies depending on the total amount of components in different designs, thus not completely meeting requirement (i). Nevertheless, to implement design with the de Bruijn graph based method, it is usually complicated and expensive, because the components need to be arranged with many crossed connections. Therefore, though de Bruijn graph has been used in various VLSI designs, it is not the mainstream choice.[74,75]

4.2 Distributed networks

4.2.1 Connection among wireless sensors

Wireless sensor network consists of a large amount of connected wireless sensors to gather information for various purposes.[76] It was conceptualized in the 1990s for the purpose of information gathering, and after decades of development, there have been many practical applications, such as environmental emergency alerting, hospital patient monitoring and livestock animal analysis.[77] Compared to wired network, the wireless sensor network can be deployed and expanded much more efficiently and freely. To design a quality wireless sensor network, some typical requirements should be met.[78,79] (i) Since the amount of wireless sensors is large, power consumption of their communications should be optimized. That is, the shortest distance between the most distanced sensors should be small. (ii) One sensor's failure may not much affect the others.

Based on the above requirements, the wireless sensor network was first modeled as a k-dimensional de Bruijn graph in 1994.[80] Similar to section 4.1.1, this is because (1) the diameter or shortest distance between the most distanced nodes in the graph is k relatively small, and (2) with a few nodes removed, most remaining nodes are still connected. In specific applications, the de Bruijn graph can be used in various levels of design. For example, one could either design

sub-networks of the complete network, or connection of the sub-networks with the de Bruijn graph.[81,82] An illustration is provided in SI Appendix, Fig. S3, where the network consists of multiple sub-networks and each sub-network is designed with a de Bruijn graph.

The wireless sensor network could be alternatively modeled as mesh, ring, tree or star structures.[98] These design methods are much simpler than the de Bruijn graph based method, but may not fully meet the requirements (i) and (ii). Specifically, in the mesh and ring structures, the most distanced sensors are across the diagonal and semicircle, respectively, which are relatively far away from each other, not meeting requirement (i); in the tree structure, a single sensor is mostly connected with only two other sensors, and in the star structure, a central sensor is connected with all the remaining sensors, so their failure will result in disconnection of the other sensors, not meeting requirement (ii). Nevertheless, similar to section 4.1.1, disadvantage of the de Bruijn graph based method is, it is not easy for implementation, so its utility is somewhat limited.[82]

4.2.2 Network design with distributed hash table

Distributed hash table (DHT) manages data stored in various nodes of a distributed network.[84,85] It was initially invented in 2001 along with development of distributed networks. With the DHT, operations such as adding or querying data can be initiated from any source node, and after the corresponding destination node is calculated, routing from the source node to the destination node is needed. For efficient routing, there are several requirements upon the network.[86,87] (i) Distance between any pair of source and destination nodes needs to be as small as possible. (ii) Routing tables stored in the nodes should be as small as possible for fast search. (iii) Failure of a single or several nodes should not break connections of other nodes.

To design such a network, the *k*-dimensional de Bruijn graph over q symbols was first used in 2003.[86] The reasons are as below. (1) The de Bruijn graph has a small graph diameter of *k*, indicating any two nodes have a small distance no more than *k*. (2) Each node in the de Bruijn graph has a small number of out-neighbors q, indicating each routing table has exactly q rows. (3) By removing one or several nodes, the majority of remaining nodes are still connected in de Bruijn graph. With the de Bruijn graph in network design, Koorde,[86] optimizes size of routing tables on top of Chord,[88] and Broose[89] and D2B[90] achieve high reliability and scalability of the network, respectively. Fig. 4 shows a comparison of the routing table size between Chord and Koorde. In the figure, the directed lines are one-to-one correspondent to the possible out-neighbors in the routing tables. The solid ones are used for the routing from node 2 to 11, while the dashed ones are not used. Chord requires a routing table of size 4, while Koorde only needs a routing table of size 2. It is worthwhile to mention that, the network with DHT is currently not an active research area, but the existing studies of de Bruijn graph's application could be referenced in other areas.

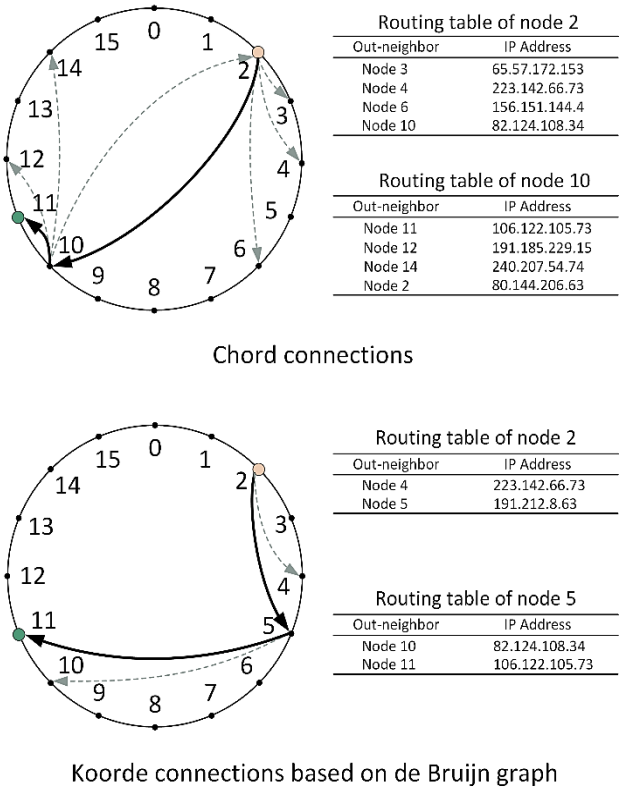

Fig. 4 Comparison of routing table size between Chord and Koorde, using routing from node 2 to node 11 as an example.

5. Variations of de Bruijn graph in above applications

De Bruijn graph in the above applications can be improved for enhanced performance. We categorize these improvements as variation in nodes, variation in edges and variation in structure, and discuss the three categories below.

5.1 Variation in nodes

In section 3.2.2, de Bruijn graph is used to represent *k*-mers from sequencing reads, and its traversal can generate the complete genome sequence. An assumption of this method is, each node in the de Bruijn graph corresponds to a unique *k*-mer in the genome, so that the graph traversal simply needs to access each node once. However, in real cases, there exist identical *k*-mers in different genome regions represented by the same node, and this poses great difficulty in generating complete genome sequence. In order to enhance specificity of the nodes, some studies incorporate the *k*-mer's positional information into each node,[50,53,57] while some studies

incorporate another related *k*-mer into each node.[49,61] The former variation is called positional de Bruijn graph, and the latter variation is called paired-end de Bruijn graph. Top-left portion of Fig. 5 gives an example of the positional de Bruijn graph, where the dashed line represents omitted nodes between (TTA,111) and (GCC,975). It is worthwhile to note that some studies construct de Bruijn graphs with preprocessed read data, so that the *k*-mers are different from the traditional, but these graphs are essentially not variations;[55,59] some other studies construct de Bruijn graphs with *k*-mers of various lengths, but these graphs differ too much from the traditional, so they are not included as variations either.[49,60]

In section 3.1.2, binary de Bruijn sequence generated from a binary de Bruijn graph can be used as correction code. This is because any subsequence of length *k* in the binary de Bruijn sequence is unique. However, to obtain much longer correction code, this requirement can be relaxed in real cases: any subsequence of length *k* is unique within some range of the sequence. That is, repeated subsequences are allowed, as long as distance between two repeated subsequences is greater than or equal to a specified value δ. To generate such sequences for correction code, a variation of the de Bruijn graph called the constrained de Bruijn graph is introduced.[11,12] Compared to the original de Bruijn graph, this variation has some nodes from the original de Bruijn graph duplicated, and the duplicated nodes are guaranteed to have a distance of at least δ. To distinguish the duplicated nodes, the corresponding sequence is added different prefixes. Bottom-left portion of Fig. 5 shows an example of the constrained de Bruijn graph.

5.2 Variation in edges

Also for section 3.2.2, some sequencing technologies produce extremely long reads but these reads may contain many errors.[35,62] The original de Bruijn graph constructed from these reads will thus contain many erroneous nodes and edges, and may fail to generate the complete genome. A potential solution to this issue is to remove unreliable *k*-mers containing sequencing errors, and construct the original de Bruijn graph, but this may result in a disconnected graph with missing nodes and edges. To keep the graph connected and avoid information loss, edges of the de Bruijn graph are improved by connecting nodes from distant *k*-mers and recording the distances. Such a variation of de Bruijn graph in edges is called A-Bruijn graph.[56,63,64] Middle portion of Fig. 5 gives an example of the A-Bruijn graph.

5.3 Variation in structure

In section 4.2.1, de Bruijn graph is used to design wireless sensor network with each node per sensor. To improve scalability and fault tolerance of the designed network, a variation of de Bruijn graph called hierarchical de Bruijn graph can be used.[80,83] This variation consists of multiple layers of nodes, and each node can either connect with nodes in neighboring layers or in the same layer. Additional layers can be added and existing layers can be removed from the hierarchical de Bruijn graph, so scalability is improved; if a node fails, the remaining nodes are still connected through neighboring layers, so fault tolerance is improved. Right portion of Fig. 5 shows an example of the hierarchical de Bruijn graph, where the dashed lines represent connections

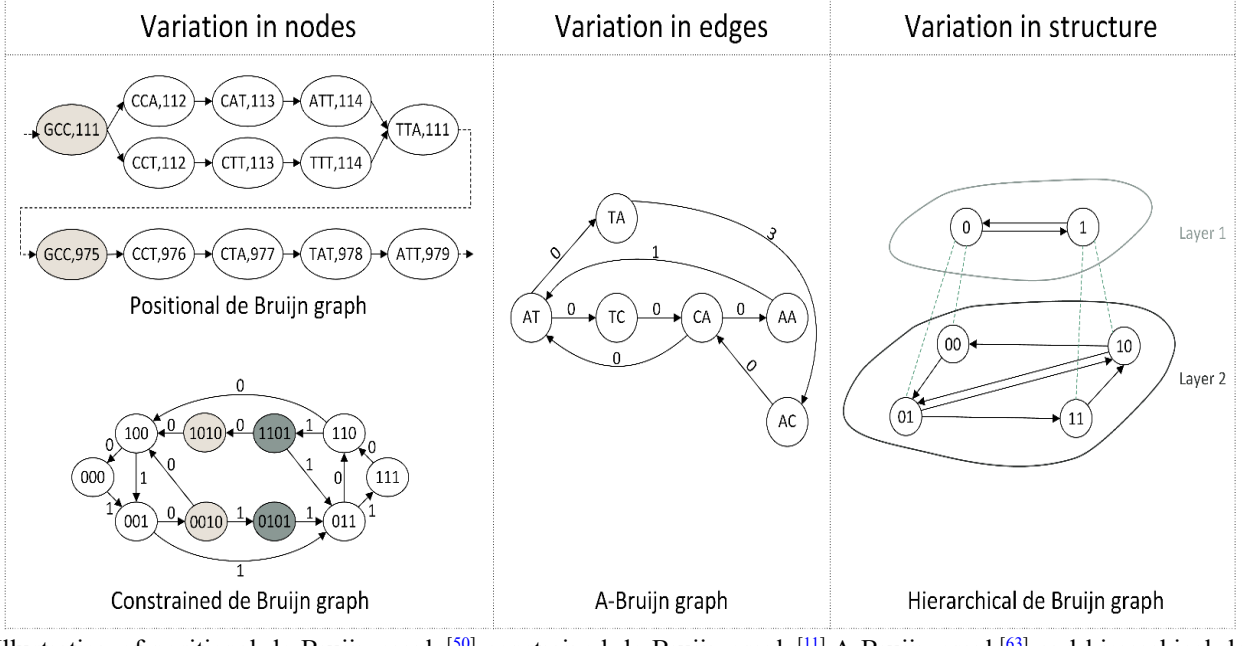

Fig. 5 Illustration of positional de Bruijn graph,[50] constrained de Bruijn graph,[11] A-Bruijn graph[63] and hierarchical de Bruijn graph.[83]

between adjacent layers.

6. Conclusion and perspectives

We present a review of the de Bruijn graph and its applications in various fields. With de Bruijn graph's node/edge features, it is used in pseudo-random sequence design, correction code design for racetrack memory, image design for self-location in tangible user interface, reversibility determination for one-dimensional cellular automata, and genome assembly; with its structure features, it is used in VLSI design, connection among wireless sensors, and network design with distributed hash table. In addition, we also summarize variations of de Bruijn graph for enhanced performance. This review honors the 105th birth anniversary of Nicolaas Govert de Bruijn and we hope it can inspire novel applications of the de Bruijn graph. At the end of this review, we provide some ideas of de Bruijn graph's novel applications.

- Firstly, the de Bruijn sequence generated with the constrained de Bruijn graph is solely used in correction code design for racetrack memory, and it would be interesting to explore its usage as pseudo-random sequence or in image design for self-location in tangible user interface.
- Secondly, the de Bruijn graph could be customized along with CA in more scientific researches, especially in the thermophysics field where CA has recently been used for simulating combustion processes or behavior of solar power systems.
- Thirdly, while genome assembly has widely adopted the de Bruijn graph method, there exist a few other applications requiring assembly of data pieces. For example, in traffic trajectory analysis, vehicle trajectories sometimes need to be assembled to recover the underlining roads, and the de Bruijn graph could be used in a similar manner to genome assembly.
- Fourthly, the de Bruijn graph has shown its applications in distributed networks, and could be further used in the broader Internet-of-things field, such as design of connections of smart home devices, cloud services and industrial components.

Brief introduction to Nicolaas Govert de Bruijn

Nicolaas Govert de Bruijn (1918-2012) was a prodigious Dutch mathematician, celebrated for his pivotal contributions to the fields of number theory, combinatorics, analysis, and logic. De Bruijn's mathematical acuity led him to discover the de Bruijn sequence and develop the de Bruijn graph, which have profound applications in a multitude of disciplines beyond mathematics, as we have summarized in our article. Additionally, de Bruijn's diverse contributions spanned various mathematical domains, including the algebraic theory for Penrose tiling, the BEST theorem, and so on. Throughout his distinguished career, Nicolaas Govert de Bruijn's remarkable contributions have profoundly shaped numerous areas of mathematics, leaving a lasting legacy for future generations to build upon.[99]

Acknowledgements

This work was funded by the National Natural Science Foundation of China (No. 62172028), and the Basic Science Center Program for Ordered Energy Conversion of the National Natural Science Foundation of China (No. 51888103).

Conflict of Interest

There is no conflict of interest.

Supporting Information

Applicable.

References

[1] N. G. De Bruijn, A combinatorial problem, *Proceedings of the Section of Sciences of the Koninklijke Nederlandse Akademie van Wetenschappen te Amsterdam*, 1946, **49**, 758-764.

[2] I. J. Good, Normal recurring decimals, *Journal of the London Mathematical Society*, 1946, **1**, 167-169, doi: 10.1112/jlms/s1-21.3.167.

[3] G. Markowsky, The sad history of random bits, *Journal of Cyber Security and Mobility*, 2014, **3**, 1-24, doi: 10.13052/jcsm2245-1439.311.

[4] S. Spinsante, M. Sarayloo, E. Gambi, C. Warty, C. Sacchi, De Bruijn sequences for DS/CDMA transmission: efficient generation, statistical analysis and performance evaluation, 2015 IEEE Aerospace Conference. Big Sky, MT, USA. IEEE, 2015, 1-11, doi: 10.1109/AERO.2015.7118890.

[5] G. Desai, S. Deshmukh, A novel de bruijn sequence-based spreading for enhancing downlink outage performance of MIMO MC-CDMA, 2017 2nd International Conference for Convergence in Technology (I2CT). Mumbai, India. IEEE, 2017, 690-694, doi: 10.1109/I2CT.2017.8226218.

[6] K. Mandal, G. Gong, Feedback reconstruction and implementations of pseudorandom number generators from composited de bruijn sequences, *IEEE Transactions on Computers*, 2016, **65**, 2725-2738, doi: 10.1109/TC.2015.2506557.

[7] G. K. Aguirre, M. G. Mattar, L. Magis-Weinberg, De Bruijn cycles for neural decoding, *NeuroImage*, 2011, **56**, 1293-1300, doi: 10.1016/j.neuroimage.2011.02.005.

[8] B. Yang, K. Mandal, M. D. Aagaard, G. Gong, Efficient composited de bruijn sequence generators, *IEEE Transactions on Computers*, 2017, **66**, 1354-1368, doi: 10.1109/TC.2017.2676763.

[9] K. Mandal, B. Yang, G. Gong, M. Aagaard, Analysis and efficient implementations of a class of composited de bruijn

sequences, *IEEE Transactions on Computers*, 2020, **69**, 1835-1848, doi: 10.1109/TC.2020.2979460.
[10] S. S. P. Parkin, M. Hayashi, L. Thomas, Magnetic domain-wall racetrack memory, *Science*, 2008, **320**, 190-194, doi: 10.1126/science.1145799.
[11] Y. M. Chee, T. Etzion, H. M. Kiah, Van Khu Vu, E. Yaakobi, Constrained de bruijn codes and their applications, 2019 IEEE International Symposium on Information Theory (ISIT). Paris, France. IEEE, 2019, 2369-2373, doi: 10.1109/ISIT.2019.8849237.
[12] Y. M. Chee, T. Etzion, H. M. Kiah, S. Marcovich, A. Vardy, Van Khu Vu, E. Yaakobi, Locally-constrained de bruijn codes: properties, enumeration, code constructions, and applications, *IEEE Transactions on Information Theory*, 2021, **67**, 7857-7875, doi: 10.1109/TIT.2021.3112300.
[13] H. Ishii, B. Ullmer, Tangible bits: towards seamless interfaces between people, bits and atoms, Proceedings of the ACM SIGCHI Conference on Human factors in computing systems. March 22 - 27, 1997, Atlanta, Georgia, USA. ACM, 1997, 234-241, doi: 10.1145/258549.258715.
[14] W. K. Bong, W. Chen, A. Bergland, Tangible user interface for social interactions for the elderly: a review of literature, *Advances in Human-Computer Interaction*, 2018, **2018**, 7249378, doi: 10.1155/2018/7249378.
[15] D. Schüsselbauer, A. Schmid, R. Wimmer, Dothraki: Tracking Tangibles Atop Tabletops Through De-Bruijn Tori, *Proceedings of the Fifteenth International Conference on Tangible, Embedded, and Embodied Interaction*, 2021, doi: 10.1145/3430524.3440656.
[16] A. M. Bruckstein, T. Etzion, R. Giryes, N. Gordon, R. J. Holt, D. Shuldiner, Simple and robust binary self-location patterns, *IEEE Transactions on Information Theory*, 2012, **58**, 4884-4889, doi: 10.1109/TIT.2012.2191699.
[17] C. J. Mitchell, K. G. Paterson, Decoding perfect maps, *Designs, Codes and Cryptography*, 1994, **4**, 11-30, doi: 10.1007/BF01388557.
[18] V. Horan, B. Stevens, Locating patterns in the de Bruijn torus, *Discrete Mathematics*, 2016, **339**, 1274-1282, doi: 10.1016/j.disc.2015.11.015.
[19] D. A. Makarov, A. D. Yashunsky, On a construction of easily decodable sub-de bruijn arrays, *Journal of Applied and Industrial Mathematics*, 2019, **13**, 280-289, doi: 10.1134/S1990478919020091.
[20] M. Gardner, The fantastic combinations of john Conway's new solitaire game "life", *Scientific American*, 1970, **223**, 20-123.
[21] O. Rozier, C. Narteau, A real-space cellular automaton laboratory, *Earth Surface Processes and Landforms*, 2014, **39**, 98-109, doi: 10.1002/esp.3479.
[22] X. Tong, Y. Feng, A review of assessment methods for cellular automata models of land-use change and urban growth, *International Journal of Geographical Information Science*, 2020, **34**, 866-898, doi: 10.1080/13658816.2019.1684499.
[23] V. R. Unni, C. K. Law, A. Saha, A cellular automata model for expanding turbulent flames, *Chaos*, 2020, **30**, 113141, doi: 10.1063/5.0018947.
[24] I. Abdennour, M. Ouardouz, A. S. Bernoussi, Modeling of electrical and thermal behaviors of photovoltaic panels using cellular automata approach. Mauri G, El Yacoubi S, Dennunzio A, Nishinari K, Manzoni L, International Conference on Cellular Automata. Cham: Springer, 2018: 57-67.10.1007/978-3-319-99813-8_5.
[25] K. Bhattacharjee, N. Naskar, S. Roy, S. Das, A survey of cellular automata: types, dynamics, non-uniformity and applications, *Natural Computing*, 2020, **19**, 433-461, doi: 10.1007/s11047-018-9696-8.
[26] J. Kari, Reversible cellular automata: from fundamental classical results to recent developments, *New Generation Computing*, 2018, **36**, 145-172, doi: 10.1007/s00354-018-0034-6.
[27] S. Wolfram, Statistical mechanics of cellular automata, *Reviews of Modern Physics*, 1983, **55**, 601-644, doi: 10.1103/revmodphys.55.601.
[28] S. Wolfram, Computation theory of cellular automata, *Communications in Mathematical Physics*, 1984, **96**, 15-57, doi: 10.1007/BF01217347.
[29] J. C. Seck-Tuoh-Mora, G. J. Martínez, Graphs related to reversibility and complexity in cellular automata, *Encyclopedia of Complexity and Systems Science. Berlin, Heidelberg: Springer*, 2017: 1-15.10.1007/978-3-642-27737-5_677-1.
[30] K. Sutner, De bruijn graphs and linear cellular automata, *Complex Systems*, 1991, **5**, 19-30.
[31] K. Sutner, Linear cellular automata and de bruijn automata, *Cellular Automata. Dordrecht: Springer*, 1999, 303-319, doi: 10.1007/978-94-015-9153-9_12.
[32] F. Dietlein, A. B. Wang, C. Fagre, A. Tang, N. J. M. Besselink, E. Cuppen, C. Li, S. R. Sunyaev, J. T. Neal, E. M. Van Allen, Genome-wide analysis of somatic noncoding mutation patterns in cancer, *Science*, 2022, **376**, eabg5601, doi: 10.1126/science.abg5601.
[33] B. Trost, B. Thiruvahindrapuram, A. J. S. Chan, W. Engchuan, E. J. Higginbotham, J. L. Howe, L. O. Loureiro, M. S. Reuter, D. Roshandel, J. Whitney, M. Zarrei, M. Bookman, C. Somerville, R. Shaath, M. Abdi, E. Aliyev, R. V. Patel, T. Nalpathamkalam, G. Pellecchia, O. Hamdan, G. Kaur, Z. Wang, J. R. MacDonald, J. Wei, W. W. L. Sung, S. Lamoureux, N. Hoang, T. Selvanayagam, N. Deflaux, M. Geng, S. Ghaffari, J. Bates, E. J. Young, Q. Ding, C. Shum, L. D'Abate, C. A. Bradley, A. Rutherford, V. Aguda, B. Apresto, N. Chen, S. Desai, X. Du, M. L. Y. Fong, S. Pullenayegum, K. Samler, T. Wang, K. Ho, T. Paton, S. L. Pereira, J.-A. Herbrick, R. F. Wintle, J. Fuerth, J. Noppornpitak, H. Ward, P. Magee, A. Al Baz, U. Kajendirarajah, S. Kapadia, J. Vlasblom, M. Valluri, J. Green, V. Seifer, M. Quirbach, O. Rennie, E. Kelley, N. Masjedi, C. Lord, M. J. Szego, M. H. Zawati, M. Lang, L. J. Strug, C. R. Marshall, G. Costain, K. Calli, A. Iaboni, A. Yusuf, P. Ambrozewicz, L. Gallagher, D. G. Amaral, J. Brian, M. Elsabbagh, S. Georgiades, D. S. Messinger, S. Ozonoff, J. Sebat, C. Sjaarda, I. M. Smith, P. Szatmari, L. Zwaigenbaum, A. Kushki, T. W. Frazier, J. A. S. Vorstman, K. A. Fakhro, B. A. Fernandez, M. E. Suzanne Lewis, R. Weksberg, M. Fiume, R. K. C. Yuen, E. Anagnostou, N. Sondheimer, D. Glazer, D. M. Hartley, S. W. Scherer, Genomic

architecture of autism from comprehensive whole-genome sequence annotation, *Cell*, 2022, **185**, 4409-4427.e18, doi: 10.1016/j.cell.2022.10.009.
[34] P. A. Cassier, R. Navaridas, M. Bellina, N. Rama, B. Ducarouge, H. Hernandez-Vargas, J.-P. Delord, J. Lengrand, A. Paradisi, L. Fattet, G. Garin, H. Gheit, C. Dalban, I. Pastushenko, D. Neves, R. Jelin, N. Gadot, N. Braissand, S. Léon, C. Degletagne, X. Matias-Guiu, M. Devouassoux-Shisheboran, E. Mery-Lamarche, J. Allard, E. Zindy, C. Decaestecker, I. Salmon, D. Perol, X. Dolcet, I. Ray-Coquard, C. Blanpain, A. Bernet, P. Mehlen, Netrin-1 blockade inhibits tumour growth and EMT features in endometrial cancer, *Nature*, 2023, **620**, 409-416, doi: 10.1038/s41586-023-06367-z.
[35] J. Eid, A. Fehr, J. Gray, K. Luong, J. Lyle, G. Otto, P. Peluso, D. Rank, P. Baybayan, B. Bettman, A. Bibillo, K. Bjornson, B. Chaudhuri, F. Christians, R. Cicero, S. Clark, R. Dalal, A. Dewinter, J. Dixon, M. Foquet, A. Gaertner, P. Hardenbol, C. Heiner, K. Hester, D. Holden, G. Kearns, X. Kong, R. Kuse, Y. Lacroix, S. Lin, P. Lundquist, C. Ma, P. Marks, M. Maxham, D. Murphy, I. Park, T. Pham, M. Phillips, J. Roy, R. Sebra, G. Shen, J. Sorenson, A. Tomaney, K. Travers, M. Trulson, J. Vieceli, J. Wegener, D. Wu, A. Yang, D. Zaccarin, P. Zhao, F. Zhong, J. Korlach, S. Turner, Real-time DNA sequencing from single polymerase molecules, *Science*, 2009, **323**, 133-138, doi: 10.1126/science.1162986.
[36] B. E. Slatko, A. F. Gardner, F. M. Ausubel, Overview of next-generation sequencing technologies, *Current Protocols in Molecular Biology*, 2018, **122**, e59, doi: 10.1002/cpmb.59.
[37] A. M. Wenger, P. Peluso, W. J. Rowell, P.-C. Chang, R. J. Hall, G. T. Concepcion, J. Ebler, A. Fungtammasan, A. Kolesnikov, N. D. Olson, A. Töpfer, M. Alonge, M. Mahmoud, Y. Qian, C.-S. Chin, A. M. Phillippy, M. C. Schatz, G. Myers, M. A. DePristo, J. Ruan, T. Marschall, F. J. Sedlazeck, J. M. Zook, H. Li, S. Koren, A. Carroll, D. R. Rank, M. W. Hunkapiller, Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome, *Nature Biotechnology*, 2019, **37**, 1155-1162, doi: 10.1038/s41587-019-0217-9.
[38] E. W. Myers, The fragment assembly string graph, *Bioinformatics*, 2005, **21**, ii79-ii85, doi: 10.1093/bioinformatics/bti1114.
[39] C.-S. Chin, P. Peluso, F. J. Sedlazeck, M. Nattestad, G. T. Concepcion, A. Clum, C. Dunn, R. O'Malley, R. Figueroa-Balderas, A. Morales-Cruz, G. R. Cramer, M. Delledonne, C. Luo, J. R. Ecker, D. Cantu, D. R. Rank, M. C. Schatz, Phased diploid genome assembly with single-molecule real-time sequencing, *Nature Methods*, 2016, **13**, 1050-1054, doi: 10.1038/nmeth.4035.
[40] S. Nurk, B. P. Walenz, A. Rhie, M. R. Vollger, G. A. Logsdon, R. Grothe, K. H. Miga, E. E. Eichler, A. M. Phillippy, S. Koren, HiCanu: accurate assembly of segmental duplications, satellites, and allelic variants from high-fidelity long reads, *Genome Research*, 2020, **30**, 1291-1305, doi: 10.1101/gr.263566.120.
[41] P. A. Pevzner, 1-Tuple DNA sequencing: computer analysis, *Journal of Biomolecular Structure and Dynamics*, 1989, **7**, 63-73, doi: 10.1080/07391102.1989.10507752.
[42] P. A. Pevzner, H. Tang, M. S. Waterman, An Eulerian path approach to DNA fragment assembly, *Proceedings of the National Academy of Sciences of the United States of America*, 2001, **98**, 9748-9753, doi: 10.1073/pnas.171285098.
[43] P. E. C. Compeau, P. A. Pevzner, G. Tesler, How to apply de Bruijn graphs to genome assembly, *Nature Biotechnology*, 2011, **29**, 987-991, doi: 10.1038/nbt.2023.
[44] D. R. Zerbino, E. Birney, Velvet: Algorithms for *de novo* short read assembly using de Bruijn graphs, *Genome Research*, 2008, **18**, 821-829, doi: 10.1101/gr.074492.107.
[45] J. T. Simpson, K. Wong, S. D. Jackman, J. E. Schein, S. J. M. Jones, I. Birol, ABySS: a parallel assembler for short read sequence data, *Genome Research*, 2009, **19**, 1117-1123, doi: 10.1101/gr.089532.108.
[46] Y. Peng, H. C. M. Leung, S. M. Yiu, F. Y. Chin, IDBA–A practical iterative de bruijn graph *de novo* assemble, Annual International Conference on Research in Computational Molecular Biology. Berlin, Heidelberg: Springer, 2010: 426-440, doi: 10.1007/978-3-642-12683-3_28.
[47] J. Butler, I. MacCallum, M. Kleber, I. A. Shlyakhter, M. K. Belmonte, E. S. Lander, C. Nusbaum, D. B. Jaffe, ALLPATHS: De novo assembly of whole-genome shotgun microreads, *Genome Research*, 2008, **18**, 810-820, doi: 10.1101/gr.7337908.
[48] S. Gnerre, I. MacCallum, D. Przybylski, F. J. Ribeiro, J. N. Burton, B. J. Walker, T. Sharpe, G. Hall, T. P. Shea, S. Sykes, A. M. Berlin, D. Aird, M. Costello, R. Daza, L. Williams, R. Nicol, A. Gnirke, C. Nusbaum, E. S. Lander, D. B. Jaffe, High-quality draft assemblies of mammalian genomes from massively parallel sequence data, *Proceedings of the National Academy of Sciences of the United States of America*, 2011, **108**, 1513-1518, doi: 10.1073/pnas.1017351108.
[49] A. Bankevich, S. Nurk, D. Antipov, A. A. Gurevich, M. Dvorkin, A. S. Kulikov, V. M. Lesin, S. I. Nikolenko, S. Pham, A. D. Prjibelski, A. V. Pyshkin, A. V. Sirotkin, N. Vyahhi, G. Tesler, M. A. Alekseyev, P. A. Pevzner, SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing, *Journal of Computational Biology*, 2012, **19**, 455-477, doi: 10.1089/cmb.2012.0021.
[50] R. Ronen, C. Boucher, H. Chitsaz, P. Pevzner, SEQuel: improving the accuracy of genome assemblies, *Bioinformatics*, 2012, **28**, i188-i196, doi: 10.1093/bioinformatics/bts219.
[51] R. Li, H. Zhu, J. Ruan, W. Qian, X. Fang, Z. Shi, Y. Li, S. Li, G. Shan, K. Kristiansen, S. Li, H. Yang, J. Wang, J. Wang, De novo assembly of human genomes with massively parallel short read sequencing, *Genome Research*, 2010, **20**, 265-272, doi: 10.1101/gr.097261.109.
[52] R. Luo, B. Liu, Y. Xie, Z. Li, W. Huang, J. Yuan, G. He, Y. Chen, Q. Pan, Y. Liu, J. Tang, G. Wu, H. Zhang, Y. Shi, Y. Liu, C. Yu, B. Wang, Y. Lu, C. Han, D. W. Cheung, S.-M. Yiu, S. Peng, X. Zhu, G. Liu, X. Liao, Y. Li, H. Yang, J. Wang, T.-W. Lam, J. Wang, SOAPdenovo2: an empirically improved memory-efficient short-read *de novo* assembler, *GigaScience*, 2012, **1**, 18, doi: 10.1186/2047-217X-1-18.
[53] E. Bao, T. Jiang, T. Girke, AlignGraph: algorithm for secondary *de novo* genome assembly guided by closely related

references, *Bioinformatics*, 2014, **30**, i319-i328, doi: 10.1093/bioinformatics/btu291.
[54] K. Mahadik, C. Wright, M. Kulkarni, S. Bagchi, S. Chaterji, Scalable genome assembly through parallel de bruijn graph construction for multiple k-mers, *Scientific Reports*, 2019, **9**, 14882, doi: 10.1038/s41598-019-51284-9.
[55] J. Ruan, H. Li, Fast and accurate long-read assembly with wtdbg2, *Nature Methods*, 2020, **17**, 155-158, doi: 10.1038/s41592-019-0669-3.
[56] M. Kolmogorov, D. M. Bickhart, B. Behsaz, A. Gurevich, M. Rayko, S. B. Shin, K. Kuhn, J. Yuan, E. Polevikov, T. P. L. Smith, P. A. Pevzner, metaFlye: scalable long-read metagenome assembly using repeat graphs, *Nature Methods*, 2020, **17**, 1103-1110, doi: 10.1038/s41592-020-00971-x.
[57] S. Huang, X. He, G. Wang, E. Bao, AlignGraph2: similar genome-assisted reassembly pipeline for PacBio long reads, *Briefings in Bioinformatics*, 2021, **22**, bbab022, doi: 10.1093/bib/bbab022.
[58] M. Rautiainen, T. Marschall, MBG: Minimizer-based sparse de Bruijn Graph construction, *Bioinformatics*, 2021, **37**, 2476-2478, doi: 10.1093/bioinformatics/btab004.
[59] B. Ekim, B. Berger, R. Chikhi, Minimizer-space de Bruijn graphs: whole-genome assembly of long reads in minutes on a personal computer, *Cell Systems*, 2021, **12**, 958-968.e6, doi: 10.1016/j.cels.2021.08.009.
[60] A. Bankevich, A. V. Bzikadze, M. Kolmogorov, D. Antipov, P. A. Pevzner, Multiplex de Bruijn graphs enable genome assembly from long, high-fidelity reads, *Nature Biotechnology*, 2022, **40**, 1075-1081, doi: 10.1038/s41587-022-01220-6.
[61] P. Medvedev, S. Pham, M. Chaisson, G. Tesler, P. Pevzner, Paired de bruijn graphs: a novel approach for incorporating mate pair information into genome assemblers, *Journal of Computational Biology*, 2011, **18**, 1625-1634, doi: 10.1089/cmb.2011.0151.
[62] M. Eisenstein, Oxford Nanopore announcement sets sequencing sector abuzz, *Nature Biotechnology*, 2012, **30**, 295-296, doi: 10.1038/nbt0412-295.
[63] Y. Lin, J. Yuan, M. Kolmogorov, M. W. Shen, M. Chaisson, P. A. Pevzner, Assembly of long error-prone reads using de Bruijn graphs, *Proceedings of the National Academy of Sciences of the United States of America*, 2016, **113**, E8396-E8405, doi: 10.1073/pnas.1604560113.
[64] C. Ye, Z. S. Ma, C. H. Cannon, M. Pop, D. W. Yu, Exploiting sparseness in *de novo* genome assembly, *BMC Bioinformatics*, 2012, **13**, S1, doi: 10.1186/1471-2105-13-S6-S1.
[65] J. S. Kilby, Invention of the integrated circuit, *IEEE Transactions on Electron Devices*, 1976, **23**, 648-654, doi: 10.1109/T-ED.1976.18467.
[66] M. S. Das, Architecture of multi-processor systems using networks on chip (noc): An overview, *CVR Journal of Science and Technology*, 2022, **22**, 7-15, doi:10.32377/cvrjst2202.
[67] J.-C. Bermond, C. Peyrat, De bruijn and kautz networks: a competitor for the hypercube? *First European Workshop on Hypercube and Distributed Computers*, 1989, 279.
[68] M. R. Samatham, D. K. Pradhan, A multiprocessor network suitable for single-chip VLSI implementation, *ACM SIGARCH Computer Architecture News*, 1984, **12**, 328-339, doi: 10.1145/773453.808202.
[69] M. R. Samatham, D. K. Pradhan, The de Bruijn multiprocessor network: a versatile parallel processing and sorting network for VLSI, *IEEE Transactions on Computers*, 1989, **38**, 567-581, doi: 10.1109/12.21149.
[70] O. Collins, S. Dolinar, R. McEliece, F. Pollara, A VLSI decomposition of the deBruijn graph, *Journal of the ACM*, **39**, 931-948, doi: 10.1145/146585.146620.
[71] H. Moussa, A. Baghdadi, M. Jezequel, Binary de Bruijn on-chip network for a flexible multiprocessor LDPC decoder, 2008 45th ACM/IEEE Design Automation Conference. Anaheim, CA, USA. IEEE, 2008, 429-434.
[72] M. Hosseinabady, M. R. Kakoee, J. Mathew, D. K. Pradhan, Low latency and energy efficient scalable architecture for massive NoCs using generalized de bruijn graph, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2011, **19**, 1469-1480, doi: 10.1109/TVLSI.2010.2050914.
[73] Y. Chen, J. Hu, X. Ling, T. Huang, A novel 3D NoC architecture based on De Bruijn graph, *Computers and Electrical Engineering*, 2012, **38**, 801-810, doi: 10.1016/j.compeleceng.2011.11.016
[74] N. Prasad, P. Mukherjee, S. Chattopadhyay, I. Chakrabarti, Design and evaluation of ZMesh topology for on-chip interconnection networks, *Journal of Parallel and Distributed Computing*, 2018, **113**, 17-36, doi: 10.1016/j.jpdc.2017.10.011.
[75] N. Habibi, M. R. Salehnamadi, A. Khademzadeh, A novel 3d mesh-based noc architecture for performance improvement, *Majlesi Journal of Electrical Engineering*, 2022, **16**, 85-101, doi: 10.30486/mjee.2022.696497.
[76] J. M. Kahn, R. H. Katz, K. S. J. Pister, Next century challenges: mobile networking for "Smart Dust", Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking, 1999, 271-278, doi: 10.1145/313451.313558.
[77] D. Kandris, C. Nakas, D. Vomvas, G. Koulouras, Applications of wireless sensor networks: an up-to-date survey, *Applied System Innovation*, 2020, **3**, 14, doi: 10.3390/asi3010014.
[78] M. H. Anisi, G. Abdul-Salaam, M. Y. I. Idris, A. W. A. Wahab, I. Ahmedy, Energy harvesting and battery power based routing in wireless sensor networks, *Wireless Networks*, 2017, **23**, 249-266, doi: 10.1007/s11276-015-1150-6.
[79] S. Chouikhi, I. El Korbi, Y. Ghamri-Doudane, L. A. Saidane, A survey on fault tolerance in small- and large-scale wireless sensor networks, *Computer Communications*, 2015, **69**, 22-37, doi: 10.1016/j.comcom.2015.05.007.
[80] S. S. Iyengar, D. N. Jayasimha, D. Nadig, A versatile architecture for the distributed sensor integration problem, *IEEE Transactions on Computers*, 1994, **43**, 175-185, doi: 10.1109/12.262122.
[81] A. A. Taleb, J. Mathew, D. K. Pradhan, Clustered de bruijn based multi layered architectures for sensor networks, International Conference on Wireless and Mobile Networks.

Berlin, Heidelberg: Springer, 2010: 123-136, doi: 10.1007/978-3-642-14171-3_11.
[82] C. Lu, D. Hu, A fault-tolerant routing algorithm for wireless sensor networks based on the structured directional de bruijn graph, *Cybernetics and Information Technologies*, 2016, **16**, 46-59, doi: 10.1515/cait-2016-0019.
[83] T. T. Huynh, C. S. Hong, A novel hierarchical routing protocol for wireless sensor networks, *International Conference on Computational Science and Its Applications. Berlin, Heidelberg: Springer Berlin Heidelberg*, 2005, 339-347, doi: 10.1007/11424758_36.
[84] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, A scalable content-addressable network, Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications. San Diego California USA. ACM, 2001, doi: 10.1145/383059.383072.
[85] S. Cherbal, A. Boukerram, A. Boubetra, A survey of DHT solutions in fixed and mobile networks, *International Journal of Communication Networks and Distributed Systems*, 2016, **17**, 14-42, doi: 10.1504/ijcnds.2016.077938.
[86] M. F. Kaashoek, D. R. Karger, Koorde: A simple degree-optimal distributed hash table, Peer-to-Peer Systems II: Second International Workshop, IPTPS 2003, Berkeley, CA, USA, February 21-22, 2003: 98-107, doi: 10.1007/978-3-540-45172-3_9.
[87] A. Datta, S. Girdzijauskas, K. Aberer, On de Bruijn routing in distributed hash tables: there and back again, Proceedings of Fourth International Conference on Peer-to-Peer Computing, 2004. Proceedings. Zurich, Switzerland. IEEE, 2004, 159-166, doi: 10.1109/PTP.2004.1334943.
[88] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan, Chord, *ACM SIGCOMM Computer Communication Review*, 2001, **31**, 149-160, doi: 10.1145/964723.383071.
[89] A.-T. Gai, L. Viennot, Broose: a practical distributed hashtable based on the de-Bruijn topology, Proceedings of Fourth International Conference on Peer-to-Peer Computing, 2004. Proceedings. Zurich, Switzerland. IEEE, 2004, 167-174, doi: 10.1109/PTP.2004.1334944.
[90] P. Fraigniaud, P. Gauron, D2B: A de Bruijn based content-addressable network, *Theoretical Computer Science*, 2006, **355**, 65-79, doi: 10.1016/j.tcs.2005.12.006.
[91] A. M. Alhakim, A simple combinatorial algorithm for de bruijn sequences, *The American Mathematical Monthly*, 2010, **117**, 728, doi: 10.4169/000298910x515794.
[92] Z. Liu, T.-Y. Sung, Routing and transmitting problems in de Bruijn networks, *IEEE Transactions on Computers*, 1996, **45**, 1056-1062, doi: 10.1109/12.537129.
[93] R. A. Rowley, B. Bose, Fault-tolerant ring embedding in de Bruijn networks, *IEEE Transactions on Computers*, 1993, **42**, 1480-1486, doi: 10.1109/12.260637.
[94] A. Vahid, G. Mappouras, D. J. Sorin, R. Calderbank, Correcting two deletions and insertions in racetrack memory, 2017: arXiv: 1701.06478, http://arxiv.org/abs/1701.06478.
[95] T. Strang, A. Dammann, M. Roeckl, S. Plass, Using gray codes as location identifiers, Ge- ographisches Institut Der Universita¨t Heidelberg Proceedings, 2009, 135-143.
[96] A. Q. Ansari, V. Sharma, R. Mishra, A 3-disjoint path design of non-blocking shuffle exchange network by extra port alignment, *The Journal of Supercomputing*, 2022, **78**, 14381-14401, doi: 10.1007/s11227-022-04450-2.
[97] A. M. Sllame, M. Daddsh, Designing torus and HyperCube network-on-chip systems based on MPLS networking technique, 2023 IEEE 3rd International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA). Benghazi, Libya. IEEE, 2023, 570-575, doi: 10.1109/MI-STA57575.2023.10169637.
[98] N. Heydarishahreza, S. Ebadollahi, R. Vahidnia, F. J. Dian, Wireless sensor networks fundamentals: a review, 2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON). Vancouver, BC, Canada. IEEE, 2020, 1-7, doi: 10.1109/IEMCON51383.2020.9284873.
[99] J. W. Klop, Nicolaas Govert de Bruijn (1918–2012) Mathematician, computer scientist, logician, *Indagationes Mathematicae*, 2013, **24**, 648-656, doi: 10.1016/j.indag.2013.09.004.