ARCHIWUM INFORMATYKI
TEORETYCZNEJ I STOSOWANEJ
Tom xx (200x), z. x
pp xxx-xxx

The Hive Network and its Routing Algorithm

IRENEUSZ SZCZEŚNIAK [a]

[a]Institute of Theoretical and Applied Informatics
Polish Academy of Sciences
ul. Bałtycka 5, 44-100 Gliwice, Poland
email: iszczesniak@iitis.gliwice.pl

Abstract: We present the hive network, a new three-dimensional network, and furnish it with an efficient routing algorithm. The network is built by interconnecting several identical honeycomb networks placed next to each other like real honeycombs in a hive. The mathematical measure of a network's complexity, hereinafter referred to as "the complexity", is defined as the product of the node degree and the length of the network diameter. Such defined complexity is important because it is related to the real-money cost. The proposed hive network is studied since it is less complex than the three-dimensional honeycomb network. The complexity of the hive network with N nodes is approximately $k_1 = 10.48\sqrt[3]{N}$, while the complexity of a three-dimensional honeycomb network is approximately $k_2 = 14.52\sqrt[3]{N}$, 38% greater. Our simulations verified that the routing algorithm generates correct results.

1. Introduction

As the demand for computational power increases, new designs for parallel computing are needed. The inherent part of the design is the topology of the interconnection network between processing units and the routing algorithm for the network. Many topologies have been proposed, as surveyed in [1] or [2].

Desired are those networks which are less complex than the others. The mathematical measure of complexity, hereinafter referred to as "the complexity", is defined as the product of the network's diameter length (i.e. the length of the longest path from the set of the network's shortest paths) and the node degree (i.e. the number of links a node has). Such defined complexity is important because it is related to the real-money cost. A network is effective in terms of complexity if for a given number of nodes its complexity is small.

The honeycomb network (also hereafter referred to as the honeycomb), described and provided with a routing algorithm in [3], is less complex than the mesh network. The honeycomb and its torus were further researched in [4], [5], [6], [7] for topological properties and communication algorithms. Several honeycomb-based networks and their properties can be found in [8]. In [9] a three-dimensional honeycomb network is proposed along with a routing algorithm and properties. The three-dimensional honeycomb network has a shorter diameter for a given number of nodes than the honeycomb network.

In the article we propose the hive network (to which we also refer simply as a hive), which is even less complex than the three-dimensional honeycomb network. For our network we develop an addressing scheme and an efficient routing algorithm. The addressing scheme for the hive network employs the addressing scheme proposed in [3] for a honeycomb. We use properties defined in [3] for nodes of a honeycomb: the color, the sign, and we use their routing algorithm for a honeycomb.

2. Hive Network

The hive network of size t is composed of $2t-1$ interconnected honeycomb networks of size t. A sample hive network of size $t=2$, which is composed of three horizontally positioned honeycomb networks of size $t=2$, is shown in Figure 1. Neighbouring honeycombs are connected with additional vertical links. All links in the hive are bidirectional.

A node in the hive network is addressed by four integer coordinates: x, y, z, v. In Figure 1 each node is labeled with its address (x, y, z, v). We address nodes in a single honeycomb with three coordinates x, y, z according to the addressing scheme for honeycombs. The v coordinate designates a honeycomb in the hive. The middle honeycomb has $v=0$, the honeycombs above the middle honeycomb have v of subsequent positive integers, while the honeycombs below the middle honeycomb have v of subsequent negative integers.

In the hive, a node either does not have a vertical link (i.e., every second node at the top or bottom of a hive network) or has only one vertical link. If a node has a vertical link, then it leads to a node either one honeycomb up or down, which depends on the color of the node. Each node in the hive network can be either white or black (see Figure 1). These colors inform about the availability of vertical links. If a node has a vertical link going down, then the node is white, while a node with a vertical link going up is black. Such defined property of nodes in a hive is hereinafter referred to by *3D color of a node* as opposed to the color property of node in a honeycomb.

We note that the color property of a node in a honeycomb is different than the 3D color property of a node in a hive. Nodes in honeycombs with the coordinate v of

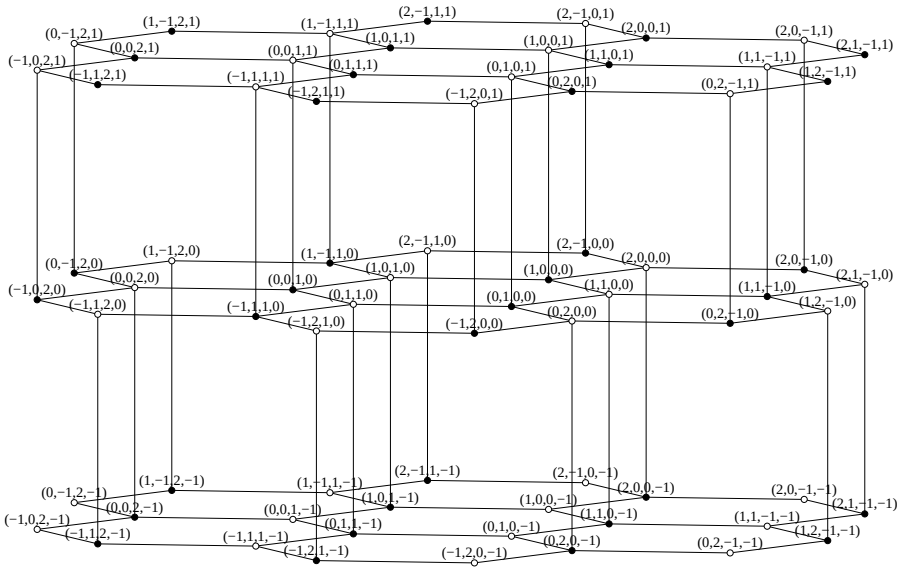

Fig. 1. Addresses of nodes in a hive network.

even values have the same colors as nodes in the honeycomb network, while nodes in honeycombs with the coordinate v of odd values have the opposite colors to the colors they would have in the honeycomb network.

2.1. Topological Properties

In this section we provide basic topological properties of the hive network and the three-dimensional honeycomb, and show that the former is less complex. Table 1 lists the formulas for the number of nodes and the diameter length as functions of the network size t. The node degree of both networks is four.

We derived the complexity of the hive network as a function of the number of nodes N in the following way. First, we obtained the formula for the network size t as a function of the number N, i.e. equation $N = (2t-1)6t^2$ was solved with respect to t (for this purpose the Mathematica package was used [10]). Second, the obtained solution was plugged into the expression for the diameter length, i.e. $6t-3$. Finally, the resulting expression for the diameter length was multiplied by four, the node degree, to yield the equation (1) for the complexity k_1 of the hive network.

The complexity k_2 of the three-dimensional honeycomb network is expressed by the equation (2), which was derived like the equation (1).

network	number of nodes N	diameter length k
hive of size t	$(2t-1)6t^2$	$6t-3$
3D honeycomb of size t	$\frac{32t^3-2t}{3}$	$8t-4$

Table 1. Topological properties of hives and three-dimensional honeycombs.

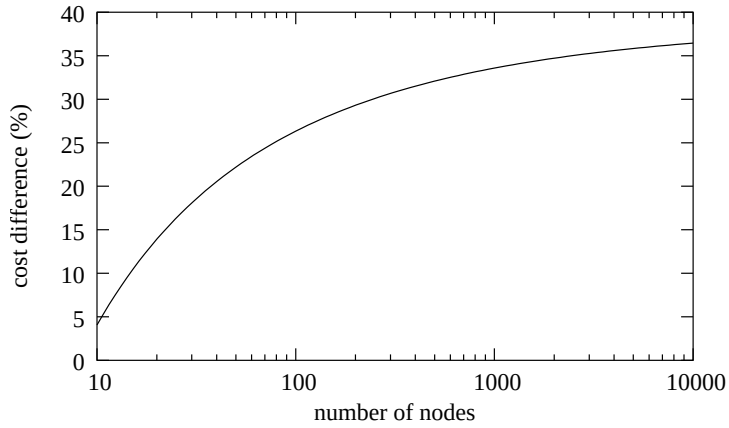

Fig. 2. Difference in complexity between a three-dimensional honeycomb and a hive.

$$k_1 = \; 4\Big(1+9\,N+3\,\sqrt{N(9\,N+2)}\Big)^{-\frac{1}{3}} + 4\Big(1+9\,N+3\,\sqrt{N(9\,N+2)}\Big)^{\frac{1}{3}} - 8 \tag{1}$$

$$k_2 = 4\left(\frac{2\left(3^{\frac{1}{3}}+\left(27\,N+\sqrt{729\,N^2-3}\right)^{\frac{2}{3}}\right)}{3^{\frac{2}{3}}\left(27\,N+\sqrt{729\,N^2-3}\right)^{\frac{1}{3}}}-4\right) \tag{2}$$

Figure 2 shows by what percent the complexity of the three-dimensional honeycomb network is greater in comparison with the complexity of the hive network. For large values of N, $k_1 \approx 10.48\sqrt[3]{N}$ and $k_2 \approx 14.52\sqrt[3]{N}$, and therefore the three-dimensional honeycomb network can be more expensive even by 38% for large values of N.

3. Preliminaries

The following are some basic definitions needed for the description of the routing algorithm:

a node address - a vector of four integers,

x, y, z, v - coordinates of a node; these symbols are used exclusively for this purpose,

$c[0]$**,** $c[1]$**,** $c[2]$**,** $c[3]$ - coordinates x, y, z, v of the node c, respectively,

$r\_part(c)$ - a function ascribing to every node c an integer from 1 to 6, which represents the Roman part (Roman parts are defined below) for details of the node (1 denotes the I part, ..., 6 denotes the VI part); "r_part" stands for "Roman part",

$color(c)$ - a function which returns 1 if the color of the node c in a honeycomb is black, otherwise returns 2,

$sign(c)$ - a function which returns 1 if the color of the node c in a honeycomb is black, otherwise returns -1,

$3d\_color(c)$ - a function which returns 1 if the 3D color of the node c is black, otherwise returns 2,

$3d\_sign(c)$ - a function which returns 1 if the 3D color of the node c is black, otherwise returns -1,

current node - a node at which a message is currently being routed, denoted by c,

next node - a node to which a message is sent from the current node, denoted by n,

destination node - a node to which a message is to be delivered, denoted by d.

For the use of the routing algorithm, a honeycomb network must be divided into six parts, I, II, III, IV, V and VI (hence the name "Roman parts") by lines which go across the network, each line partitioning the network into halves according to the sign of one of nodes' coordinates. A sample division is shown in Figure 3. Checking every node's coordinate x, y, z for its sign (by the condition of $coordinate > 0$ or $coordinate \leq 0$), a node can be ascribed to one of the six parts. Table 2 contains conditions which coordinates of a given node must meet to be classified to a particular Roman part.

4. Routing Algorithm

The algorithm NEXT-NODE (Algorithm 1) is the routing algorithm. It calculates the address of the next node n in the shortest path between the current node c and the destination node d. The algorithm is given the addresses of the nodes c and d, and the size t of the network. The succeeding paragraphs describe three phases into which the algorithm can be logically split. The beginning of each phase is marked with a comment in the listing of the algorithm.

In the course of the first phase it is checked whether a message can be sent between honeycombs, which is viable if there is a need to send a message between honeycombs

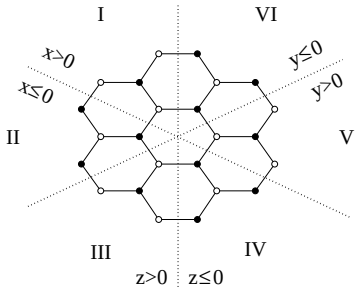

Fig. 3. Partition to Roman parts.

part	condition
I	$x > 0 \wedge y \leq 0 \wedge z > 0$
II	$x \leq 0 \wedge y \leq 0 \wedge z > 0$
III	$x \leq 0 \wedge y > 0 \wedge z > 0$
IV	$x \leq 0 \wedge y > 0 \wedge z \leq 0$
V	$x > 0 \wedge y > 0 \wedge z \leq 0$
VI	$x > 0 \wedge y \leq 0 \wedge z \leq 0$

Table 2. Conditions for the Roman parts.

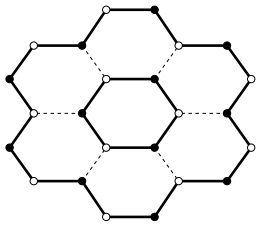

Fig. 4. Circles of links in a honeycomb.

(values of the v coordinate of the nodes c and d differ), and at the current node c there is an appropriate vertical link available. If the address of the next node n is found in this phase, the algorithm returns it and quits, otherwise the second phase is performed.

In the second phase the next node n is searched for in the honeycomb containing the node c. Here the routing algorithm for a honeycomb is used. Given the address of the current node c and the address of the destination node d, the algorithm calculates the address of the next node n in the shortest path between the nodes c and d provided that the next node n is in the same level as the current node c. The address of the next node n is obtained only if the values of the coordinates v, y, z of the nodes c and d differ. If the address of the next node n is found in this phase, the algorithm returns it and quits, otherwise the third phase is performed.

The third phase is carried out for the nodes c and d, whose addresses differ only by the values of the coordinate v, and there is no appropriate vertical link available from the node c. Sending a message in this case requires a roundabout path. Links of roundabout paths form circles in a honeycomb, as shown in Figure 4. In this phase a roundabout link is chosen. Every node in a honeycomb has two links belonging to a circle: the clockwise link and the counterclockwise link, and the choice between them is arbitrary. Here we use counterclockwise links. In the algorithm the matrix $\mathbf{A}$ is used, which contains indexes of coordinates of the node c to be modified.

4.1. Message Complexity of the Algorithm

Since the algorithm sends a message along the shortest path, the length of any routing path is at most equal to the diameter length of the hive network. The message complexity of a routing algorithm is defined as the number of times a message is sent along links during its delivery. Because the diameter length of the hive network, expressed by equation (1), is of the order $\sqrt[3]{N}$, therefore the message complexity of the algorithm is $O(\sqrt[3]{N})$.

5. Conclusions

In the article the hive network is proposed along with an addressing scheme and a routing algorithm. The algorithm has been successfully verified with simulation software. We show that the complexity of the three-dimensional honeycomb, our contender, is greater even by 38% than the complexity of the hive network.

It must be noted that the three-dimensional honeycomb is more regular and its routing algorithm more elegant than the hive network and its routing algorithm. Nonetheless, we believe that the symmetry and elegance should be sacrificed for smaller complexity and better performance. This is an example where symmetry and elegance are not always beneficial.

Acknowledgement

I thank sincerely prof. Zbigniew Czech and dr. Wojciech Mikanik for advice, and my wife Arlena for assistance with English.

References

[1] I. Stojmenović: *Direct interconnection networks*, Parallel and Distributed Computing Handbook (A.Y. Zomaya, ed.), McGraw-Hill, Inc., 1996, pp. 537-567.

[2] G. Kotsis: *Interconnection Topologies and Routing for Parallel Processing Systems*, Technical Report Series, ACPC/TR 92-19, 1992.

[3] I. Stojmenović: *Honeycomb Networks: Topological Properties and Communication Algorithms*, IEEE Transactions on Parallel and Distributed Systems, 8(10): 1036-1042, 1997.

[4] J. Carle, J.F. Myoupo, D. Seme: *All-to-all broadcasting algorithms on honeycomb networks and applications*, Parallel Processing Letters, Vol. 9, No. 4 (1999), pp. 539-550

[5] G.M. Megson, X. Liu, X. Yang: *Fault-tolerant ring embedding in a honeycomb torus with node failures*, Parallel Processing Letters, Vol. 9, No. 4 (1999), pp. 551-561

[6] G.M. Megson, X. Yang, X. Liu: *Honeycomb tori are Hamiltonian*, Information Processing Letters, v. 72 n. 3-4, pp. 99-103, Nov. 1999

[7] B. Parhami, D.M. Kwai: *A Unified Formulation of Honeycomb and Diamond Networks*, IEEE Trans. Parallel Distrib. Systems 12(1)(January 2001), pp. 74-79

[8] W. Mikanik: *Three-dimensional Variants of Honeycomb Networks*, technical report, Silesian University of Technology, Gliwice, Poland, December 1999.

[9] J. Carle, J.F. Myoupo, I. Stojmenović: *Higher Dimensional Honeycomb Networks*, Journal of Interconnection Networks, Vol. 2, No. 4 (2001), pp. 391-420.

[10] `http://www.wolfram.com/mathematica`

Sieć w kształcie ula i jej algorytm wyznaczania trasy

Streszczenie

W artykule zaproponowano nową trójwymiarową sieć połączeń w kształcie ula dla komputerów wieloprocesorowych, oraz jej algorytm wyznaczania trasy przesyłania pakietów. Sieć jest zbudowana przez połączenie wielu identycznych sieci plastra miodu, które są umieszczone obok siebie jak prawdziwe plastry miodu w ulu. Badanie tej sieci jest uzasadnione, ponieważ jej koszt jest mniejszy w porównaniu z trójwymiarową siecią plastra miodu. Matematyczna miara kosztu jest definiowana jako iloczyn długości średnicy sieci i stopnia węzłów. Tak zdefiniowana miara przekłada się na koszt wyrażony w pieniądzu. Matematyczny koszt proponowanej sieci z liczbą węzłów N wynosi w przybliżeniu $k_1 = 10.48\sqrt[3]{N}$, natomiast koszt trójwymiarowej sieci plastra miodu wynosi w przybliżeniu $k_2 = 14.52\sqrt[3]{N}$, 38% więcej. Symulacyjnie sprawdzono poprawność wyników zwracanych przez algorytm wyznaczania trasy.

Algorithm 1 NEXT-NODE(t, c, d)

$\mathbf{A} = (a_{ij}) = \begin{pmatrix} 2 & 1 & 1 & 0 & 0 & 2 \\ 0 & 0 & 2 & 2 & 1 & 1 \end{pmatrix}$
{first phase: try to send a message between honeycombs}
$n \Leftarrow c$
if $(d[3] - c[3])3d\_sign(c) > 0$ and $|n[3] + 3d\_sign(c)| < t$ **then**
  $n[3] \Leftarrow n[3] + 3d\_sign(c)$
  return n
end if
{second phase: try to send a message using the routing algorithm for a honeycomb}
$n \Leftarrow c$
for $i \Leftarrow 0$ to 2 **do**
  if $(d[i] - c[i])sign(c) > 0$ **then**
    $n[i] \Leftarrow n[i] + sign(c)$
    return n
  end if
end for
{third phase: use a roundabout path}
$j \Leftarrow r\_part(c)$
for $i \Leftarrow 1, 2$ **do**
  if $(d[a_{ij}] - c[a_{ij}])sign(c) > 0$ **then**
    $n[a_{ij}] \Leftarrow n[a_{ij}] + sign(c)$
    return n
  end if
end for
$i \Leftarrow color(c)$
$n[a_{ij}] \Leftarrow n[a_{ij}] + sign(c)$
return n