J Supercomput (2007) 39:73–89
DOI 10.1007/s11227-006-0005-4

Single-row mapping and transformation of connected graphs

Shaharuddin Salleh · Stephan Olariu · Albert Y. Zomaya · Kiew Leh Yieng · Nur Arina B. Aziz

Abstract In a dimensional problem, the transformation of a graph into its linear network can be viewed as a transition involving demand and supply. A connected graph represents the demand flows between the components in the graph while the network supporting it is the resource or capacity links for supporting the demand volumes. The transformation involves a mapping from the graph to its network to satisfy certain performance metrics. In this work, we propose a model for transforming a connected graph to its linear network model in the form of a single-row routing network. The main objective is to provide an optimum routing network that minimizes the congestion. In this technique, the given graph is first partitioned into several disjoint cliques using the Hopfield neural network using our model called AdCliq. From the cliques, a set of intervals derived from the zones are obtained through the matching nodes in the single-row axis. The intervals are then mapped into a network of non-crossing nets using our previously developed tool called ESSR. The network is optimal in terms of minimum street congestion and number of doglegs, and this provides a reasonably good step towards the overall solution to the demand-supply problem.

Keywords Connected graph · Single-row routing · Demand-supply problem and linear transformation

1 Introduction

In most cases, a connected graph represents a network of inter-dependent nodes where their links provide the communication paths between the nodes. In a connected graph, a node can

S. Salleh (✉) · K. L. Yieng · N. A. B. Aziz
Dept. of Mathematics, Faculty of Science, Universiti Teknologi Malaysia, 81310 Johor Bahru, Malaysia

S. Olariu
Computer Science Dept., Old Dominion University, Norfolk, VA 23529, USA

A. Y. Zomaya
School of Information Technologies, University of Sydney, Sydney, NSW 2006, Australia

communicate with any other node by hopping through one or more intermediate nodes. In a multi-commodity network problem, the nodes in a connected graph represent the demand and they are competing against each other for the available resources, or link capacities, in the form of network equipment (or hardware). The objective is to minimize an objective function subject to some constraints. A linear programming model is obtained if the equations in the objective function and its constraints form a linear system.

In this paper, two important components are discussed and linked to each other. First is the connected graph which is an abstraction representing a demand in the demand-supply problem. Second is single-row routing which represents the supply in the problem. Traditionally, single-row routing is a classical problem that contributes in the printed-circuit board (PCB) designs [1]. We study the demand-supply relationship between a connected graph with the single-row routing problem, and show how the latter can be applied for solving the problem in the former.

In [2], we discussed a technique for transforming a complete graph into its single-row representation. A complete graph is a fully-connected graph where every node is adjacent to all other nodes in the graph. Very often, many applications in science and engineering are reducible to this type of graph. Hence, a simplified form of a complete graph contributes in providing the solutions to these problems. In this paper, we present a technique for transforming a complete graph into a single-row routing problem. Single-row routing is a classical technique in the VLSI design that is known to be NP-complete. We solved this problem earlier using a method called ESSR, and, the same technique is applied to the present work to transform a complete graph into its single-row routing representation.

Our work in this paper is motivated from the need to expand the scope of the single-row routing problem. In [2], we showed that single-row routing can also be applied in two non-PCB applications, namely, the channel assignment problem in the wireless cellular network systems and the theoretical single-row multiprocessor network system. In the first application, each node in the given graph maps into a zone consisting of several channels for serving the demand from the mobile users. In the second application, the nodes in the single-row axis are modeled as processors in a theoretical multiprocessor system. Therefore, it is safe to say that single-row routing technique can also be applied to cover many other problems, particularly those that require matchings between the pairs of nodes in the graph.

It is obvious that two NP-complete problems are encountered here, namely, the maximum clique of a graph and single-row routing. In supporting our work, two tools have been developed, both using C++. First is AdCliq which is a visual simulator for partitioning a connected graph into sets of cliques using the Hopfield neural network method. Second is ESSR [3] which generates a realization in the form of single-row representation of the nodes grouped in cliques in AdCliq. ESSR performs a search on the minimum energy function of the nets based on the stochastic simulated annealing method. As a team AdCliq and ESSR form a strong combination for transforming the nodes in a connected graph into their single-row realization.

In this paper, we propose a technique for transforming a connected graph into its single-row representation. The method starts by partitioning the graph into several disjoint cliques using the Hopfield neural network. Once the cliques have been established, the step continues by forming an interval graph with matching nodes in the single-row axis. Finally, we apply ESSR to obtain its least congested single-row routing.

The paper is organized into eight sections. Section 2 is the problem formulation while the terminologies and symbols used are outlined in Section 3. Section 4 discusses some brief introduction to the single-row routing problem, and its solution using ESSR. The maximum clique problem and its solution using the Hopfield neural network is discussed in Section 5.

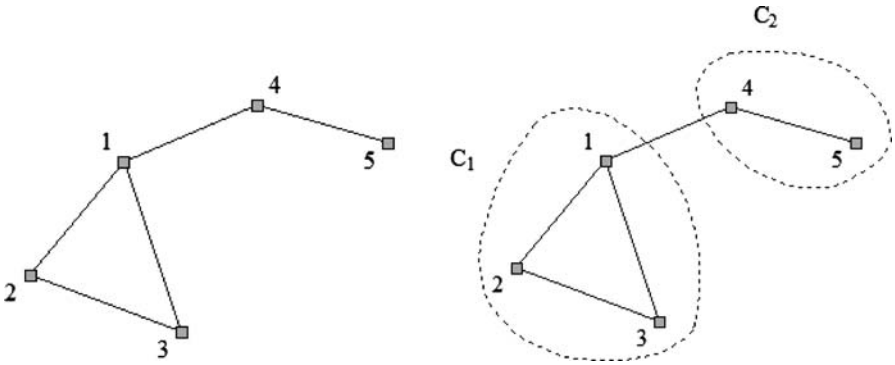

Fig. 1 A connected graph G_5 with five nodes (left) and its partitions into two cliques (right)

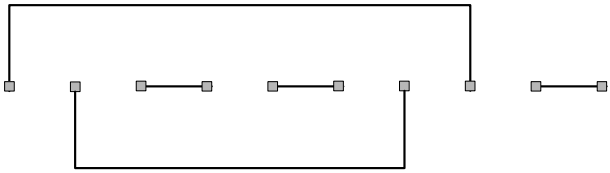

Fig. 2 Single-row representation S from G

This is followed by a description of the transformation technique using both AdCliq and ESSR in Section 6. We then display some experimental results from the simulations using AdCliq and ESSR in Section 7. Finally, Section 8 is the summary and conclusion.

2 Problem formulation

The problem can be stated as follows: given a connected graph G with an arbitrary number of nodes and edges, how can this graph be transformed into a network of non-crossing nets with their proper matching nodes? It is also our main objective in the problem to design a routing in the network with minimum street congestion and number of doglegs.

The above problem may be expressed as a *dimensional problem* in the form of demand-supply problem [4]. In this case, the graph represents the demand while the single-row network is the supply or equipment (hardware) for supporting the demand. Therefore, the problem translates into mapping the demand to the equipment in such away to meet certain performance metrics. In many cases, the demand-supply problem can be expressed as a constrained or unconstrained linear programming problem.

Figure 1 shows an example of a connected graph G_5 with five nodes. Our strategy in solving the problem requires transforming G_5 into two cliques, C_1 and C_2, before mapping the nodes into a single-row network S with 10 pins. The transformation requires a series of steps including partitioning the graph into several cliques before getting the linear representation.

Figure 2 show theresult of the transformation into its single-row representation. The figure shows the final realization of the single-row representation with a minimum street congestion value of 1 and having no dogleg. This realization is optimal with its energy at the minimum value of 2.

3 Symbols and terminologies

Several symbols and terminologies related to a connected graph and its transformation to the single-row model are explained as follows:

Graph and hopfield network models

G	Graph
G'	Set of all computed cliques in G
C_i	Clique i in G'
G_L	List of nodes in G that are not in the computed cliques
G_M	G with an additional node v_0
M	Number of nodes in G
H	Energy in the Hopfield network
R	Transmission range for connectivity in G
d_i	Degree of node i in G
q_{ij}	Element in the adjacency matrix of G
u_{xi}	Input of neuron i in group x
v_i	Node i in G
v_{xi}	Output of neuron i in group x
w_{ij}	Weight of link between v_i and v_j

Single-row network model

E	Energy in the realization
Q	Congestion in the realization
Q_u	Upper street congestion in the realization
Q_l	Lower street congestion in the realization
D	Number of doglegs in the realization
N	Number of nets in S
L	Ordering list of nodes in the single-row axis S
I_k	Interval k
S	Set of pins in the single-row axis
b_k	beginning pin in net k
e_k	ending pin in net k
s_i	Pin i in S
n_k	Net k in S
$h_{k,j}$	Height of segment j in net k

4 Single-row routing problem

Single-row routing is a combinatorial optimization problem that has been proven to be NP-complete [1]. Traditionally, single-row routing is one of the techniques employed for designing the routes between the electronic components of a printed-circuit board. Each path joining the pins is called a *net*. In single-row routing problem, we are given a set of $2m$ evenly-spaced pins (terminals or vias), t_i, for $i = 1, 2, \ldots, 2m$, arranged horizontally from

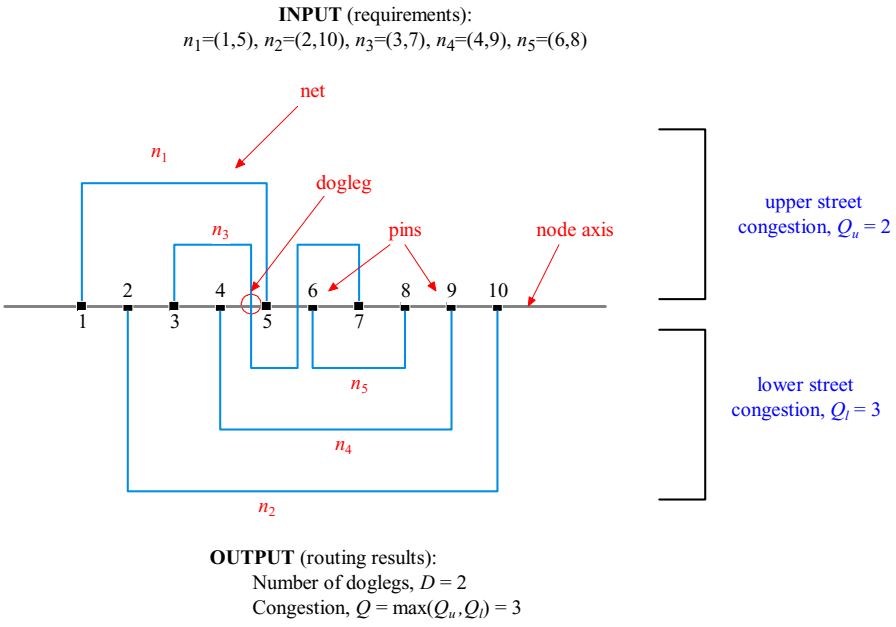

Fig. 3 Terminologies in the single-row routing problem

left to right in a single horizontal row called *single-row axis*. The problem is to construct m nets from the list $L = \{n_k\}$, for $k = 1, 2, \ldots, m$, formed from horizontal intervals, (b_k, e_k), in the node axis, where b_k and e_k are the beginning and end pins of the intervals, respectively. Each horizontal interval is formed from a pair of two (or more) pins through non-intersecting vertical and horizontal lines. The nets are to be drawn from left to right, while the reverse direction is not allowed.

Figure 3 shows a realization in a single-row routing from the ordering list $L = \{n_1, n_3, n_5, n_4, n_2\}$. Physically, each net in the single row represents a conductor path for its pins to communicate. The area above the single-row axis is called the *upper street*, while that below is the *lower street*. The number of horizontal tracks in the upper and lower streets is called the *upper street congestion* Q_u and the *lower street congestion* Q_l, respectively. The overall street congestion Q of a realization is defined as the maximum of its upper and lower street congestions, that is, $Q = \max(Q_u, Q_l) = 3$ in the above figure. A crossing on the node axis, as shown through a line between node 4 and 5 in the figure, is called a *dogleg* or interstreet crossing. The realization also produces two doglegs in this example.

4.1 Simulated annealing approach using ESSR

Various methods based on graph theory, mathematical programming and heuristics have been proposed to solve the single-row routing problem. In [5], Kuh *et al.* proposed some necessary and sufficient conditions for determining the minimum congestion. In [6], a graph decomposition technique was proposed to obtain the least congested routing based on the overlapping intervals of a graph. In Deogun and Sherwani [7], Du and Liu proposed a heuristic for finding an optimal routing based on a method that sorts the nets according to their classes, internal cut numbers and residual cut numbers.

One important objective in single-row routing is to minimize both the street congestion and number of doglegs. This objective is difficult to achieve as the two components are separate but dependent entities. While having one component minimized, the other tends to show some resistance to its minimization. In [8], we proposed the simulated annealing approach for solving the single-row routing problem. In this work, the energy in a given net ordering is expressed as the total length of all the tracks, according to the energy function E as follows:

$$E = \sum_{k=1}^{M} \sum_{j=1}^{M_k} |h_{k,j}|. \tag{1}$$

In the above equation, $h_{k,j}$ is the height of segment j in net k, while M is the number of nets in the problem and M_k is the number of segments in net k. The routing produced in Fig. 3 with $L = \{n_1, n_3, n_5, n_4, n_2\}$ has an energy given by $E = 11$ which suggests it may not be optimum. A better solution can be obtained by reforming the list with different orderings of nets.

ESSR produces reasonably good optimal results in terms of both minimum street congestion and number of doglegs through a series of iterative steps. The idea in ESSR is to place the nets in a list L in order according to their position. Starting at a high temperature the process starts with a random list L_0 where the energy E_0 is recorded. The temperature is then lowered gradually where, at the same time, the position of a set of nets in the list are swapped. The new energy is recorded and its difference from the old energy ΔE determines whether the new list is accepted or rejected. The new list is accepted if $\Delta E \leq 0$. If $\Delta E > 0$ then the new list if accepted only if its Boltzmann probability given by $P(\Delta E) = e^{-\Delta E/T}$ is greater than some threshold value ε. This annealing step is repeated until the energy is minimum, and that no further improvement is noted after several iterations. The list corresponding to this minimum energy is then the solution to the problem, and this list produces the desired least-congested routing.

The method is further improved in [3] where a set of nets, rather than just one pair, are swapped to produce a faster convergence to its solution. This parallel version of simulated annealing is called the *enhanced simulated annealing technique for single-row routing*, or ESSR. Through comparison with some methods before, ESSR produces reasonably good optimal results in terms of both the street congestion and number of doglegs.

5 Hopfield network approach to the maximum clique problem

A clique C of a graph G is a subgraph of G where a node in C is adjacent to every other node in the subgraph. The number of nodes in a clique is called its cardinality. A clique with the maximum cardinality in a graph is also called the maximum clique. The maximum clique problem is one of the most fundamental problems in graph theory which has many applications such as in scheduling, clustering and resource allocation. The problem is known to be NP-complete as it has interacting variables with many degrees of freedom.

The maximum clique problem is one of the most fundamental problems in graph theory. Several methods have been proposed to solve this problem. One such method is the Hopfield neural network [9] where an advantage can be seen from the fact that the graph maps directly to the network. The network itself is a dynamic system which allows the neurons to update their values asynchronously on new inputs, and maintain these values as inputs to other

neurons. This process leads the network to enter a new state which depends on some iterative parameters such as time and the thermostatic temperature. The network is said to stabilize when changes on the input values for some neurons bring no effect to the network as a whole.

Hopfield neural network is a fully-connected, recurrent network, which has strong potential for hardware implementation, due to the collective dynamical properties of its interacting neurons [9]. The network itself is a complete graph where a node is connected to every other node in the network. Updates on the nodes are executed asynchronously where only a few nodes are selected at random at each time step. Unlike feed-forward networks, the Hopfield network involves cyclic connections that make them behave as a nonlinear dynamic system. Essentially, the system has very rich temporal and spatial behaviors, such as stable and unstable fixed points, limit cycles, and chaotic behavior which can be used to model associative memory. The network dynamics are dominated by locally stable states called *attractors*, from where their state equations converge after a long temporal evolution.

We implement the method discussed in [11] which formulates the maximum clique problem as an unconstrained quadratic zero-one problem, with the energy function $f(x)$ given as follows:

$$f(\mathbf{x}) = \mathbf{b^T x} + \mathbf{x^T Q x} = \sum_{i=1}^{M} b_i x_i + \sum_{i=1}^{M}\sum_{j=1}^{M} q_{ij} x_i x_j, \tag{2}$$

for the vector $\mathrm{x} = [x_1, x_2, \ldots, x_M]^T$ with $x_i \in \{0, 1\}$, $\mathrm{b} = [b_1, b_2, \ldots, b_M]^T$ and $\mathbf{Q} = [q_{ij}]$ is an $M \times M$ symmetrical rational matrix with zero diagonal elements. The corresponding objective here is to minimize $f(\mathbf{x})$.

A new matrix G_M is obtained by adding v_0 into G and this makes the unconstrained quadratic zero-one programming problem equivalent to the problem of minimizing the weight summation over the same partition in G_M. From this concept, the maximum clique problem becomes the global minimization of the following energy function [11]:

$$H = \sum_{x=0}^{M}\sum_{y=0}^{M}\sum_{i=1}^{2} w_{xy} v_{xi} v_{yi} \tag{3}$$

where $\sum_{i=1}^{2} v_{xi} = 1 \text{for} x = 0, 1, \ldots, M$ where $v_{xi} \in \{0, 1\} \text{and} v_{01} = 1.$In this case, the weight w_{ij} of an edge between nodes i, j in G_M is defined by

$$\begin{aligned} w_{0i} &= w_{i0} = \frac{1}{4}\left(\sum_{j=1}^{N} q_{ij} - 1\right), \\ w_{ii} &= 0; w_{ij} = w_{ji} = \tfrac{1}{4} q_{ij} \end{aligned} \tag{4}$$

for all $i \neq j \text{and } i, j = 1, \ldots, M$.

In general, the M-node maximum clique problem can be mapped onto the Hopfield network with M neurons. Comparing the Hopfield energy function (2) with the energy function of Equation (3) defined for the maximum clique problem, we obtain all the self-connection weights, $w_{xi,xi} = 0$, the interconnection between neurons in the same group x, $w_{xi,xj} = 0$ and the bias for every neuron $\theta_{xi} = 0$. Substituting these values in Hopfield's updating rule

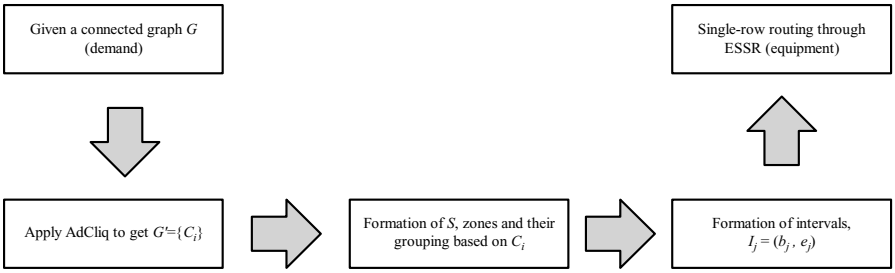

Fig. 4 The steps in the transformation

(3), we obtain the input of neuron i in the xth group given by

$$u_{xi} = -2 \sum_{y=0}^{M} w_{xy} v_{yi}. \tag{5}$$

The dynamics of the network are reduced to

$$v_{xi}(t+1) = \begin{cases} 1 & \text{if } u_{xi}(t) = \max\limits_{j=1,\ldots,m} \{u_{xj}(t)\}, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

We chose the group of neurons, x with the smallest energy difference to be updated at time t, which is $\Delta H_x(t) = \min_{y=1,\ldots,n}\{\Delta H_y(t)\}$. This selection enables the network to converge to a stable state in just a few iterations, and the global minimum leads to the maximum clique of the graph.

6 Single-row transformation

Single-row transformation of a graph involves four steps, as shown in Fig. 4. In the first step, the given connected graph G is partitioned into several disjoint cliques $G' = \{C_i\}$. The given graph represents the demand in the demand-supply problem. Our approach for finding the cliques is based on the Hopfield neural network method through AdCliq, as described in Section 4.1. The second step involves the formation of nodes on a single-row axis and their classification into zones. This leads way to the third step which is the formation of intervals for matching the nodes on the single-row axis. The last step produces the optimal realization for the minimum congestion in the network using ESSR. The solution is expressed in the form of a network realization which represents the equipment in the demand-supply problem.

6.1 AdCliq: Our Tool for partitioning a graph into cliques

In our work, the maximum clique problem is an important component in transforming a given graph G to its linear form for the demand-supply problem. An important step in the work is to partition G into q disjoint cliques $G = \{C_1, C_2, \ldots, C_q\}$ in such a way that $|C_1| \geq |C_2| \geq \cdots \geq |C_i| \geq \cdots \geq |C_q|$, where $|C_i|$ is the cardinality of C_i. In this notation,

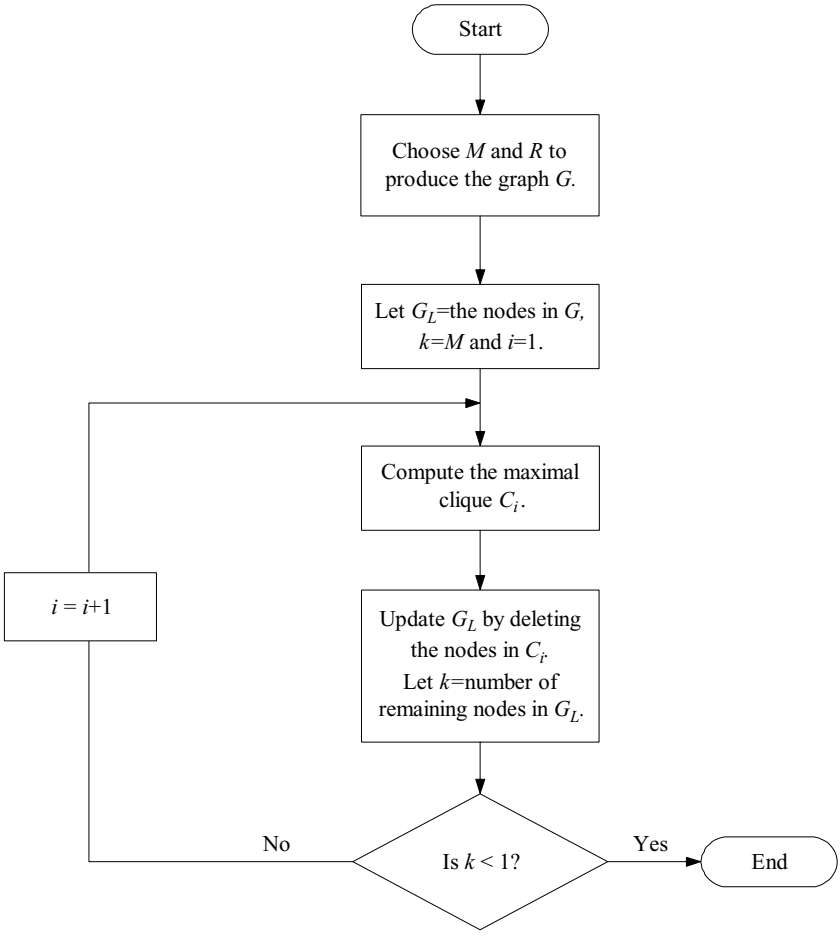

Fig. 5 Execution steps in AdCliq

C_1 is the maximum clique of G, C_2 is the next maximum clique with the nodes in C_1 excluded from G, and continues until the minimum clique C_q.

Our model is called AdCliq, and it is based on the Hopfield neural network. AdCliq has been developed using the C++ programming language on the Windows environment. The simulation model generates a graph G with two to 100 nodes at the randomly determined positions in the window. A link or edge between a pair of nodes is determined through their Euclidean distance: a link exists if their Euclidean distance is less than a preset threshold value R. This approach provides links to the nodes of the graph that are close to each other, and ignore those that are far apart from each other. Therefore, an area that is congested with nodes is said to have a high density, and this prompts for a higher number of links to meet the higher demand of communication and data sharing.

Figure 5 showsthe flowchart of the execution steps in AdCliq. The model starts with input in the form of M and R, for the number of nodes in G and the transmission range, respectively. From the input, a list G_L of all the nodes in G is created, with n as its number. Initially, $k = M$ but this value will decrease as G_L excludes all the nodes from the computed cliques.

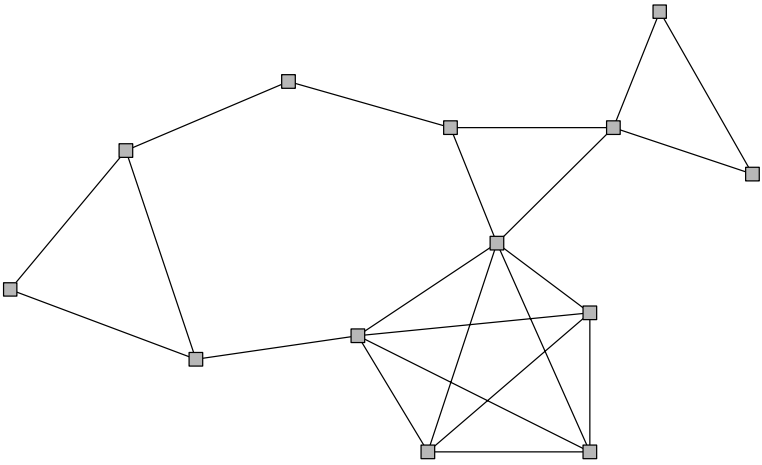

Fig. 6 A graph with 13 nodes

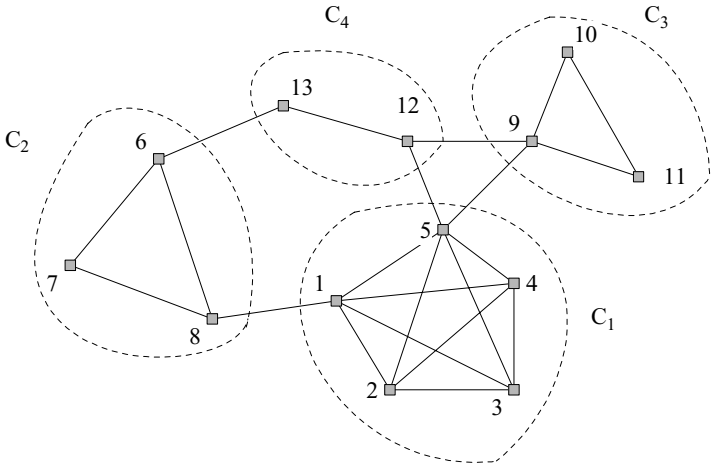

Fig. 7 Cliques of the graph in Fig. 6

The next step in AdCliq is to compute the maximum clique based on the list of nodes in G_L. The clique is denoted C_1, and once it has been determined all the nodes in C_1 are removed from G_L. The number of nodes k in G_L is updated, and a check is made to determine whether the iteration continues or to be stopped. If $k < 1$ the operation is stopped as it signals all the cliques in G have been computed. Otherwise, the above step is repeated for finding the second maximum clique, or C_2, and so on until the stopping criteria $k < 1$ is reached.

Adcliq is illustrated using an example on the graph in Fig. 6. The connected graph G consists of 13 nodes, labeled as v_i where $i = 1, 2, \ldots, 13$. Applying AdCliq, we obtain $G' = \{C_1, C_2, C_3, C_4\}$, with $C_1 = \{v_1, v_2, v_3, v_4, v_5\}$, $C_2 = \{v_6, v_7, v_8\}$, $C_3 = \{v_9, v_{10}, v_{11}\}$ and $C_4 = \{v_{12}, v_{13}\}$, as shown in Fig. 7.

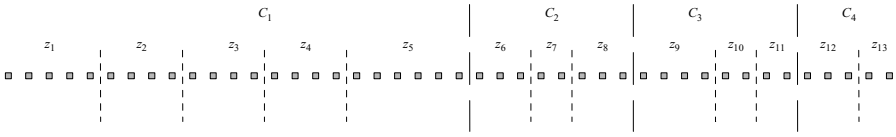

Fig. 8 Transformation of G' into S

6.2 Formation of zones in the single-row node axis

The second step involves the formation of the single-row node axis S from G. In the transformation, the graph G with M nodes generates N nets in S, where each net represents a pair of nodes in S. The node v_i in G forms zone z_i consisting of d_i nodes in S, where d_i is the degree of v_i. It can easily be shown that there are $2N = \frac{1}{2}\sum_{i=1}^{m} d_i$ nodes in S. The zones are further classified as cliques C_j according to the graph G'.

The next step is to form a set of linear nodes S from G'. Each node v_i in G' expands into d_i nodes in S, where d_i is the degree of v_i. Figure 8 shows the linear nodes from G' obtained from Fig. 7. In the figure, $d_1 = 5$ since the degree of v_1 is 5. There are $\sum_{i=1}^{13} d_i = 44$ nodes in S, and they are placed into 13 zones, z_i , for $i = 1, 2, \ldots, 13$, according to their origin in G'. The nodes in S are labeled s_k, for $k = 1, 2, \ldots, 44$.

It is obvious from the transformation that a node in G becomes a zone in S. The zones z_i in Fig. 8 are arranged according to their grouping into cliques, in the order from C_1 to C_4. This is necessary to make sure the neighboring nodes are close to each other in the new arrangement. It is also necessary to minimize the length of each interval to be formed in the subsequent steps as the length of an interval is also the horizontal length of the net.

6.3 Formation of intervals

The third step involves matching the pins $S = \{s_k\}$ into pairs of two to form the intervals I_k, for $j = 1, 2, \ldots, N$. Each interval $I_k = (b_k, e_k)$ is made up of a left (beginning) pin b_k and a right (ending) pin e_k. The formation of intervals is based on the width and level of the intervals, similar to the method discussed in our earlier work in [2].

The matching between the nodes in G' can be either intra-clique or inter-clique. A matching is said to be *intra-clique* if the two nodes belong to the same clique. Otherwise, it is *inter-clique*. In both cases, the matchings represent the nets to be drawn from left to right in S.

Intra-clique matching involves a technique as described in Algorithms 1 and 2 in [2]. In this technique, the nets are formed by grouping the intervals into several layers based on their width. The *width* of interval k, denoted by w_k, is defined as the difference between its beginning and end pins, given as $w_k = e_k - b_k$. A *level, y,* in S_m consists of a set of equal-width nets grouped in ascending order from the smallest width to the largest. Our strategy begins by forming levels where the nets with equal width are grouped. The nets in S_q are created by defining their end-points. Once the nets have been formed, the next step consists of sorting and renumbering the nets based on their beginning pins, in ascending order from the lowest to highest.

The ith net in level y in S_q, denoted as $n_{y,i,q} = (b_{y,i,q}, e_{y,i,q})$, formed from the clique, C_q, is grouped into levels based on its width, $w_{y,q}$, according to the following relationships:

$$b_{y,i,q} = (q - y) + (q - 1)(i - 1) \tag{9a}$$

$$e_{y,i,q} = b_{y,i} + w_{y,q} \tag{9b}$$

Table 1 Matching pairs obtained from the transformation

net	pins
n_1	(1,31)
n_2	(2,21)
n_3	(3,16)
n_4	(4,11)
n_5	(5,6)
n_6	(7,20)
n_7	(8,15)
n_8	(9,10)
n_9	(12,19)
n_{10}	(13,14)
n_{11}	(17,18)
n_{12}	(22,41)
n_{13}	(23,32)
n_{14}	(24,44)
n_{15}	(25,30)
n_{16}	(26,27)
n_{17}	(28,29)
n_{18}	(33,40)
n_{19}	(34,39)
n_{20}	(35,36)
n_{21}	(37,38)
n_{22}	(42,43)

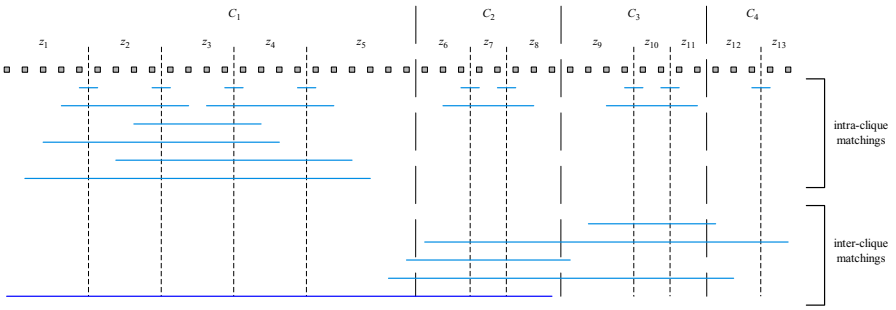

Fig. 9 Interval graph formation from the matching nodes

for $y = 1, 2, \ldots, q-1$, and $i = 1, 2, \ldots, q-1$.

Inter-clique matchings involve the creation of interval $I_k = (b_k, e_k)$ formed from a pin in z_i and another in z_j from the graph relationship (v_i, v_j) where v_i and v_j are not in the same clique in G'. Figure 9 shows the intervals formed from $G' = \{C_1, C_2, C_3, C_4\}$ in the inter-clique matching. The results from the inter-clique matchings are tabulated in Table 1.

6.4 Transformation of G into a single row S and its routing

In the last step, each interval I_k in S is mapped and realized into net n_k using ESSR. The net is drawn from left to right in such a way that it is made up of horizontal and vertical line segments only, and it does not cross another net. We apply ESSR with $I = \{I_k\}$ as the input, to produce an optimal routing with minimum congestion based on the minimum energy E,

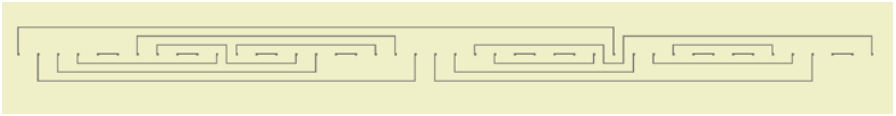

Fig. 10 Final realization with $E = 26$, $Q = 3$ and $D = 3$, using ESSR

as described in our earlier work in [1]. Figure 10 showsthe routing results from the nets in Table 1. The results are optimal with $E = 26$, $Q = 3$ and $D = 3$ from the ordering list $L = \{1, 14, 19, 5, 20, 6, 9, 21, 10, 7, 16, 8, 4, 17, 11, 18, 22, 15, 13, 3, 12, 2\}$.

7 Experimental results and analysis

Our simulations are performed using AdCliq and ESSR. Both AdCliq and ESSR are user-friendly tools that were developed using C++ on the Windows environment. AdCliq supports a maximum of 100 nodes per graph which are generated randomly in the rectangular area of the window. The connected graph is produced from the random placement of nodes in the plane. The adjacency of the nodes in the graph G is controlled by the preset transmission range R. In this case, a pair of nodes are adjacent (an edge exists between the nodes) if its distance is less than or equal to R. This factor contributes to the total number of links in G, which is the same as the total degrees of all its nodes, GD. R is directly proportional to GD: a high value of R contributes to a high GD, which means a high adjacency rate between the pairs of nodes in G. AdCliq computes the number of cliques of the graph, its maximum clique and the total degrees, and display the results in the window.

We illustrate the interface in AdCliq using an example of a graph with 30 nodes. Figure 11 showsthe visual interface of AdCliq with G having 30 nodes scattered randomly in the rectangular area on the left of the window. A sample run with $R = 150$ produces ten cliques in G, with the maximum clique of six and total node degrees of 160. The maximum clique produced is $C_1 = \{3, 7, 10, 15, 24, 28\}$. Other cliques are $C_2 = \{6, 18, 19, 21\}$, $C_3 = \{2, 8, 11, 27\}$,C_4 = 13,16,20,29 , $C_5 = \{5, 22, 23, 30\}$, $C_6 = \{4, 12\}$, $C_7 = \{14, 17\}$, $C_8 = \{1, 25\}$, $C_9 = \{9\}$ and $C_{10} = \{26\}$.

Figures 12 showsESSR in forming the intervals between the pins in S using the information from the graph G in AdCliq. The iterations become stable after 2270 iterations with the minimal energy value $E = 2209$, and this results in $Q = 24$ and $D = 335$. The energy E is the optimal value obtained after some massive annealing steps in ESSR which correspond to minimum Q and D based on Equation (1). The details on the energy minimization procedures in ESSR can be referred to our earlier work in [3]. This output is shown as a realization in Fig. 13.

We perform simulations using AdCliq and ESSR on 14 data sets where the above illustrations make up Set 8. The graphs with between six to 60 nodes are generated using AdCliq, while their transformation into the single rows are performed using ESSR. Table 2 shows the results from the simulation on the data sets. The graphs information using AdCliq is shown in columns 2 through 6 in the table. AdCliq computes the number of cliques, the maximum clique MC and the total number of degrees GD of each graph. It then follows with the transformation of each graph G into a row of pins using ESSR, with the information on the transformation shown in columns 7 to 11 in the table. In the transformation process, the number of nodes in G maps directly as the number of zones in S. In addition, each node degree in G maps to a pin in S. The output in the form of energy E, congestion Q and number

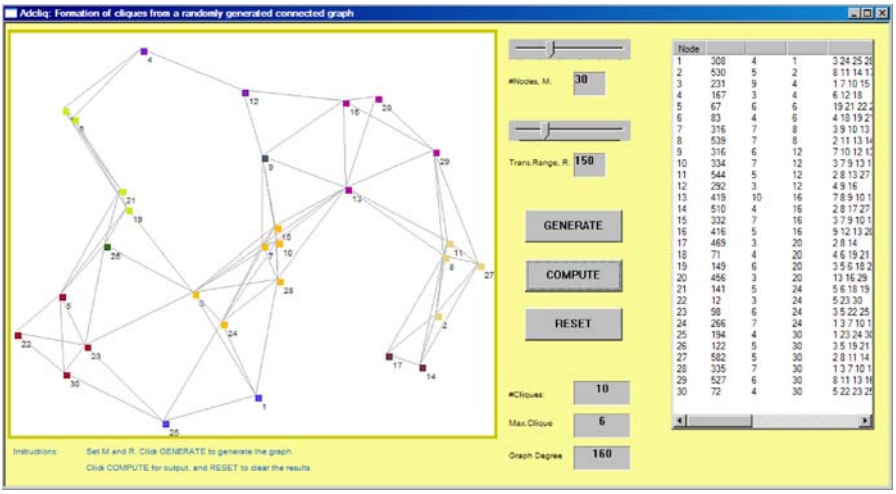

Fig. 11 AdCliq interface showing the graph in Set 8

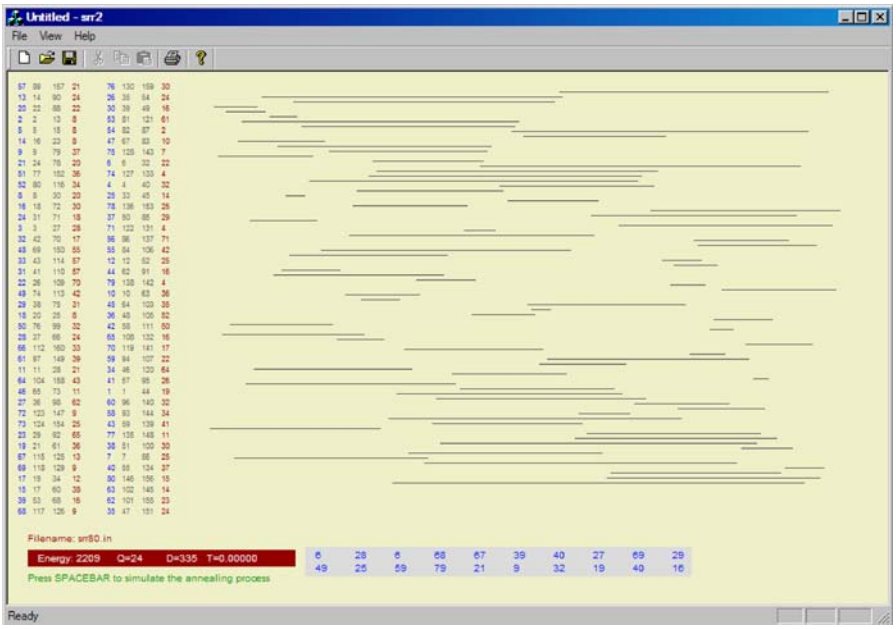

Fig. 12 Intervals *S* formed from the graph in Fig. 11

of doglegs *D* from the simulated annealing processes are shown in columns 9, 10 and 11, respectively.

Figure 14 shows the relationships between the number of nodes *M* in the graph with the energy (left), and congestion and number of doglegs (right) on the sets of data in Table 2. It is obvious from the graphs that the complexity of the problem increases exponentially as the number of nodes increases.

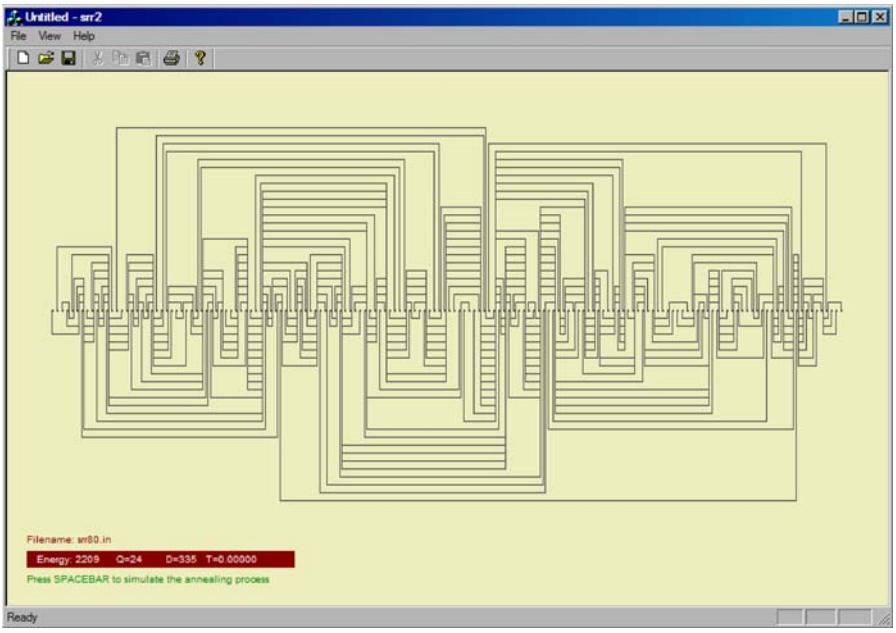

Fig. 13 Final realization in S with 160 pins

Table 2 Sample results from the simulations

		G Partitioning using AdCliq				S transformation using ESSR				
Set	M	R	#Cliques	MC	GD	#Zones	#Pins	E	Q	D
1	6	250	3	3	12	6	12	12	2	3
2	8	250	4	3	22	8	22	20	3	4
3	10	250	4	4	28	10	28	24	4	2
4	13	200	4	5	44	13	44	26	3	3
5	15	200	5	5	68	15	68	151	10	30
6	20	200	8	5	80	20	80	343	12	36
7	25	200	9	6	134	25	134	994	19	157
8	30	150	10	6	160	30	160	2209	24	335
9	35	150	12	6	204	35	204	3872	37	425
10	40	150	14	8	232	40	232	4817	48	663
11	45	150	13	9	310	45	310	6534	57	847
12	50	100	13	7	414	50	414	9590	70	994
13	55	100	14	8	556	55	556	12954	88	1134
14	60	100	15	10	634	60	634	15337	102	1529

The transformation problem has a number of open problems in the form of theoretical foundation and applications. The transformation model from a connected graph to the single row network has a huge potential for a wide range of applications. In [2], we discussed two potential applications, namely the channel assignment problem and the multiprocessor routings. Both problems may be explored further as there are many open problems relevant to the transformation model discussed in this paper.

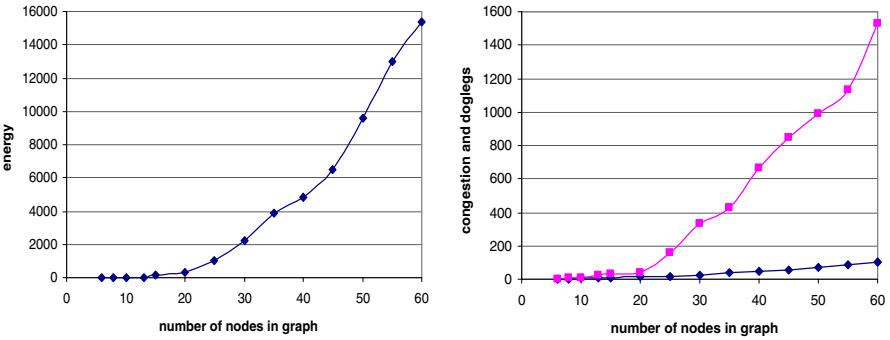

Fig. 14 Relationships between the number of nodes M with E, Q and D

We discuss briefly the potential application of the transformation model to the channel assignment problem. The case mentioned in [2] is specific to a complete graph where all nodes in the graph have equal degrees as they are connected to each other. Therefore, the transformation is made in such a way that every zone has an equal number of pins (channels). A zone in this model may represent a cell in the network. Practically, the model may not apply well as the demand for channels may differ between the cellular cells in the network. The present model is more flexible as it caters to the non-uniform channel requirements as the degree of each node in the graph varies. As shown in Fig. 9, a clique instead of a zone may represent a cell in this case. In turn, a zone may represent the mobile subscriber who may want to communicate with more than one person in the network as specified by its node degree.

8 Summary and conclusion

In this paper, the single-row transformation model of a connected graph has been presented. The transformation model is proposed as a multi-commodity problem in the form of supply-demand mapping. In the demand-supply dimensional problem, the connected graph may represent the demand flows between the components in the graph while the network supporting it is the resource or capacity links for supporting the demand volumes. The main idea behind this transformation model is to provide a reasonably good optimal single-row routing network that minimizes the congestion in the network. We proposed AdCliq which is a technique for partitioning a given graph into several disjoint cliques using the Hopfield neural network. From the cliques, a set of intervals derived from the zones are obtained through the matching nodes in the single-row axis. The intervals are then mapped into a network of non-crossing nets using our previously developed tool called ESSR. The network is optimal in terms of minimal street congestion and number of doglegs, and this provides a reasonably good step towards the overall solution to the demand-supply problem.

References

1. So HC (1974) Some theoretical results on the outing of multilayer printed wiring board. In: Proc. IEEE Symp Circuits Syst pp. 296–303
2. Salleh S, Olariu S, Sanugi B, MIA Aziz, (2005) Single-row transformation of complete graphs. J Supercomput 31(3):265–279
3. Salleh S, Sanugi B, Jamaluddin H, Olariu S, Zomaya AY (2002) Enhanced simulated annealing technique for the single-row routing problem. J Supercomput 21(3):285–302

4. Pioro M, Medhi D (2004) Routing, flow and capacity design communication and computer networks. Morgan-Kaufmann.
5. Kuh ES, Kashiwabara T, Fujisawa T (1979) On optimum single-row routing. IEEE Trans Circuits Syst 26(6):361–368
6. Deogun JS, Sherwani NA (1988) A decomposition scheme for single-row routing problems. Technical Report Series #68, Dept of Computer Science, Univ. of Nebraska.
7. Du DH, Liu LH (1987) Heuristic algorithms for single row routing. IEEE Trans Comput 36(3):312–319
8. Salleh S, Zomaya AY (1999) Scheduling for parallel computer systems: fuzzy and annealing techniques. Kluwer Academic Publishing, Boston.
9. Hopfield JJ, Tank DW (1985) Neural computation of decisions in optimization problems. Biol Cybern 52:141–152
10. Galán-Marín G, Muñoz-Pérez J (2001) Design and analysis of maximum Hopfield networks. IEEE Trans Neural Netw 12(2):329–339
11. Takefuji Y (1992) Neural network parallel computing. Kluwer Academic Publishers, Boston.