Maciej Paszyński*

h-RELATION PERSONALIZED COMMUNICATION STRATEGY

This paper considers the communication patterns arising from the partition of geometrical domain into sub-domains, when data is exchanged between processors assigned to adjacent sub-domains. It presents the algorithm constructing bipartite graphs covering the graph representation of the partitioned domain, as well as the scheduling algorithm utilizing the coloring of the bipartite graphs. Specifically, when the communication pattern arises from the partition of a 2D geometric area, the planar graph representation of the domain is partitioned into not more than two bipartite graphs and a third graph with maximum vertex valency 2, by means of the presented algorithm. In the general case, the algorithm finds $h-1$ or fewer bipartite graphs, where h is the maximum number of neighbors. Finally, the task of message scheduling is reduced to a set of independent scheduling problems over the bipartite graphs. The algorithms are supported by a theoretical discussion on their correctness and efficiency.

Keywords: *scheduling, concurrent point-to-point communications*

STRATEGIA KOMUNIKACJI OPARTA NA RELACJI h

W artykule omówiono problem szeregowania komunikacji pomiędzy procesorami przypisanymi do poddziedzin otrzymanych w wyniku podziału obszaru na podobszary, przy założeniu że dane wymieniane są pomiędzy sąsiadującymi podobszarami. W artykule przedstawiony został algorytm tworzenia grafów dwudzielnych w oparciu o grafową reprezentację obszaru podzielonego na podobszary. Przedstawiono również algorytm szeregowania bazujący na kolorowaniu skonstruowanych grafów dwudzielnych. W szczególności, kiedy rozważamy komunikację w obrębie obszarów dwuwymiarowych, graf reprezentujący podzielony obszar dwuwymiarowy jest grafem planarnym, i rozważany algorytm zdekomponuje go na dwa grafy dwudzielne oraz trzeci graf o maksymalnej walencji wierzchołka równej 2. W ogólnym przypadku (np. gdy rozważamy obszary trójwymiarowe) przedstawiony algorytm znajdzie $h-1$ lub mniej grafów dwudzielnych, gdzie h oznacza maksymalną liczbę sąsiadujących podobszarów. Zadanie szeregowanie komunikatów zostało zredukowane do niezależnych zadań szeregowania na grafach dwudzielnych. Artykuł podsumowuje analiza teoretyczna poprawności i efektywności omówionych algorytmów.

Słowa kluczowe: *szeregowanie, równoczesna komunikacja pomiędzy parami procesorów*

* Department of Computer Science, AGH University of Science and Technology, al. Mickiewicza 30, 30-059, Kraków, Poland, maciej.paszynski@agh.edu.pl

1. Introduction

Main results of the paper. In the h-relation personalized communication problem, each processor has possibly different amounts of data to share with some subset of the other processors, such that each processor is the origin or destination of at most h messages [2]. In the general case, where nothing can be assumed about the regularity of the communication pattern, the h-relation personalized communication strategy problem can be solved by using two transpose primitives [2] (all-to-all communications where each processor has to send a unique block of data to every processor, and all the blocks are of the same size).

The paper presents an alternative strategy for h-relation personalized communications (*h-rpc*) by employing bipartite graphs derived from the communication graph. Through this reformulation, the task of message scheduling is reduced to a set of independent problems of scheduling over the bipartite graphs.

We present a simple algorithm for partitioning of the planar graph into two bipartite graphs and another graph with maximum vertex valency 2. In a general case, when the communication pattern is a non-planar graph, our algorithm partitions the graph into $h-1$ bipartite graphs.

In particular, we consider the communication pattern arising from the partition of geometrical area into sub-domains, and data is exchanged between processors representing adjacent sub-domains in a geometrical sense.

We show that an efficient implementation of message passing can be represented as *h-rpc* strategy with h equal to the maximum number of neighboring sub-domains over the graph representation of the partitioned computational domain.

We include a theoretical discussion of the algorithm's correctness and an analysis of its complexities.

Routing h-relation. The *h-rpc* problem can be solved using a simple *fixed-pattern* algorithm [17]. However, if there is a large variation in the message size, the algorithm becomes inefficient. A scalable *two-phase* algorithm for the *h-rpc* was proposed by Bader, Helman, and JáJá in [2] to mitigate this inefficiency. The authors also noted that the problem can be solved by using two transpose primitives.

Some basic work with MPI primitive implementations was done by Johnsson [12, 11, 7, 8]. Efficient implementation of the MPI primitives is described in papers of Bader [1], Sundar et al. [16]; and the approach presented in [2] is acclaimed to be the optimal strategy for routing h-relation.

The authors of [2] showed that the complexity of the algorithm is $O\left(\tau + h + \left(h + \frac{n}{p} + p^2\right)\sigma\right)$, where τ is the message initialization time, n is total amount of data assumed to be distributed in $\frac{n}{p}$ portions over p processors, and σ is the time per byte at which processor can send or receive data through the network ($\frac{1}{\sigma}$ is the bandwidth of the network). In their paper, Goldman and Trystram [5] noticed that when all the communication patterns are known by every processor, a *global knowledge* algorithm can be used. By finding an optimal solution through a permutation on the communication matrix, their solution becomes potentially better; however, they used a greedy algorithm whose results can be far from the optimal schedule.

In our approach, we use a *global knowledge* communication strategy based on scheduling over bipartite graphs, with an overall complexity

$$O\left(\tau h + \frac{n}{p}\sigma\right), \tag{1}$$

which scales favorably when the processor count is increased.

Partitions of the planar graph. A *bipartite* graph is a graph whose set of vertices can be decomposed into two disjoint sets such that no two graph vertices within the same set are adjacent. A *forest* is an acyclic graph (set of possibly disconnected trees). Every forest is a bipartite graph, but not every bipartite graph is a forest. A *vertex valency* $d(x)$ is the number of edges incident to it. The maximum vertex valency in a graph G is denoted by $\Delta(G)$.

The problem of partitioning planar graphs has been extensively investigated in graph theory using forests. An effective algorithm for finding three forest partitions of every planar graph is presented in Chen et al. [19]. However, there are no theoretical results on the upper bound for the valency of forests.

The most recent theoretical results are presented in He, Hou et al. [18], and can be summarized as follows. Every planar graph G has an edge-partition in two forests T_1 and T_2 and another graph H with $\Delta(H) \leq 8$. An exact evaluation of $\Delta(H)$ is still an open problem, according to He Hou et al. [18]. There are examples of planar graphs whose graphs H have the property $\Delta(H) = 2$, which implies that $2 \leq \Delta(H) \leq 8$. The theoretical results presented in He, Hou et al. [18] do not imply any simple graph partition algorithm, since their proofs are based on mathematical induction over the number of vertices and edges in G.

We present a simple linear time algorithm for partitioning a planar graph into not more than three bipartite graphs $\{B_i\}_{i=1,2,3}$ with $\Delta(B_1) \leq h$, $\Delta(B_2) \leq h-1$ and $\Delta(B_3) \leq 2$. We use bipartite graphs that may not be forests, because we do not need to map partitions into a set of forests, we only need partitions consisting of bipartite graphs.

Scheduling over bipartite graphs. The usefulness of edge coloring for scheduling data transfers in large parallel architectures has been demonstrated by several authors, and a comprehensive survey of applications in various fields has been compiled by Jain et al. [6]. The application of simple edge-coloring algorithms to scheduling problems and evaluation of their experimental performance are presented by Marathe et al. [10, 13]. In general, when nothing can be assumed about the regularity of a graph, the problem of edge coloring is NP-complete (see [3, 4]). However, the problem of edge coloring of bipartite graphs can be solved by a polynomial-time algorithm. The fastest known bipartite graph coloring algorithm was developed by Rizzi et al. [14, 9]. The computational complexity of the algorithm is $O(n \log n \log v + e \log v)$ where n is the number of nodes, e is the number of edges, and v is the maximum vertex degree in the graph.

2. Graph partition algorithm

Graph representation of the partitioned domain. Let us consider a domain divided into sub-domains. A *graph representation of the domain* can be created,

$$G = (V, E) \tag{2}$$

where graph vertices V denote sub-domains, and edges $E = \{\{u, v\} : u, v \in V\}$ denote their adjacency. A graph G is assumed to be undirected, since the sub-domains adjacency relation is symmetric.

Partition of the graph. The *layer-connection algorithm* can be applied to the graph representation of the domain. The goal of this algorithm is to partition the graph into a set of subgraphs called *layers* and *connections* (Figs. 1 and 2).

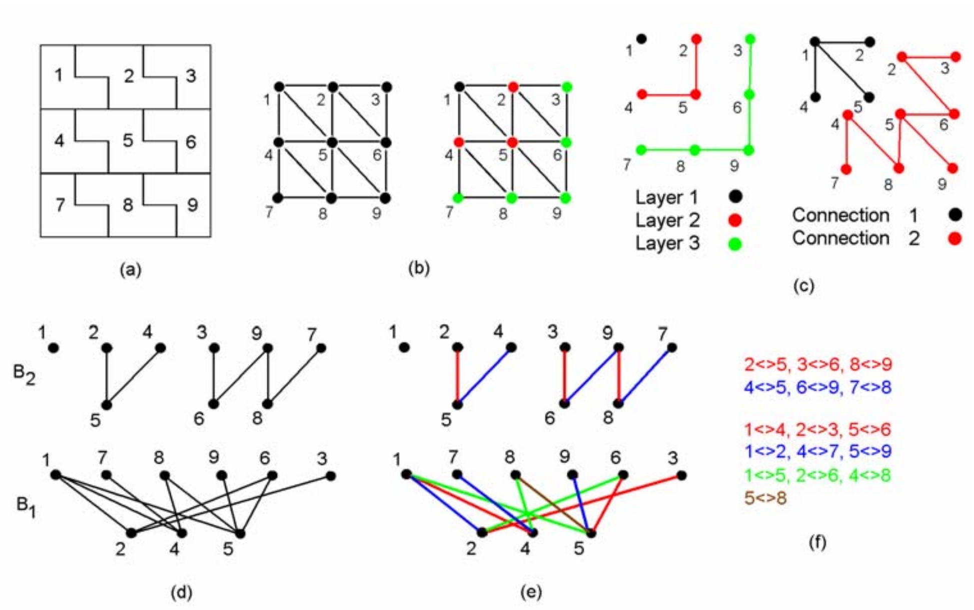

Fig. 1. Example of a nice message scheduling for partitioned $2D$ domain: **(a)** 2D domain partitioned into sub-domains. **(b)** Planar graph representation of partitioned domain. **(c)** Layer and connection subgraphs. **(d)** Partition of the planar graph into two bipartite graphs: union of all connection subgraphs B_1; union of all layer subgraphs B_2. **(e)** Edge-coloring of bipartite graphs. **(f)** Six sets of messages to be sent

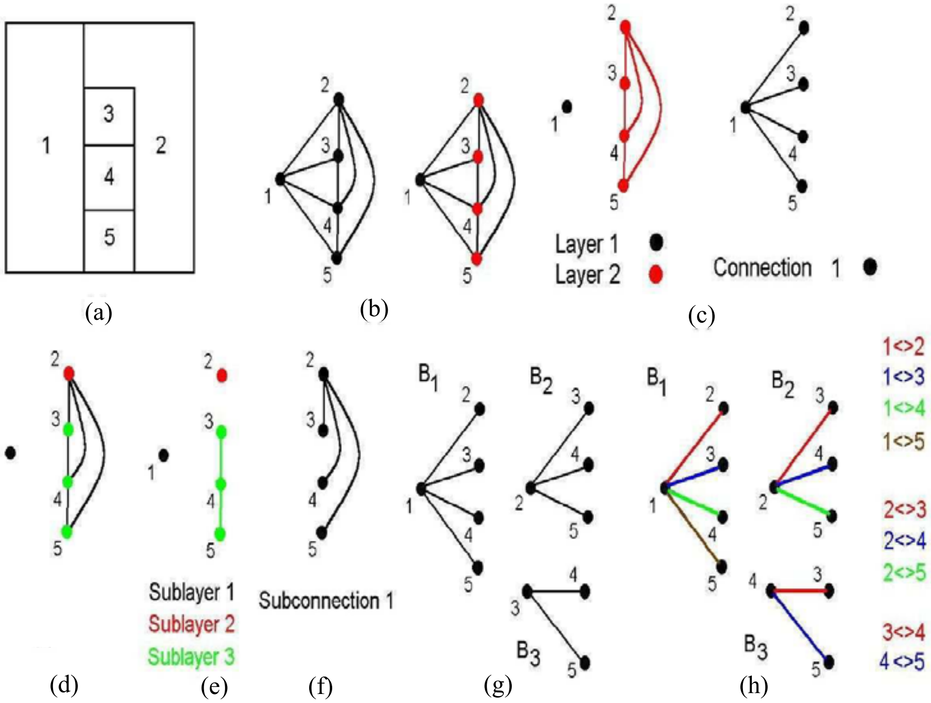

Fig. 2. Example of the worst case of message scheduling for $2D$ partitioned domain: **(a)** 2D domain partitioned into sub-domains. **(b)** Planar graph representation of partitioned domain. **(c)** Layer and connection subgraphs. **(d)** Second execution of layer-connection algorithm over layers. **(e)** Sub-layer and sub-connection subgraphs. **(f)** Partition of the planar graph into three bipartite graphs: union of all connection subgraphs B_1; union of all sub-connection sub-graphs B_2; union of all sub-layer subgraphs B_3. **(g)** Edge-coloring of bipartite graphs. **(h)** Nine messages to be sent.

A single vertex representing a sub-domain located on the boundary of the domain is selected randomly and assigned to the first layer. Then all neighbors of the previously selected vertex are assigned to the second layer. The procedure is repeated by assigning the sequence of neighboring vertices to consecutive layers until all vertices have been assigned to some layer.

More formally, we select one vertex u_0 in the graph and create an *attributing function* (3) that assigns a layer index to each graph vertex, such that within a minimal length path connecting the vertex with the initial vertex u_0, all vertices belong to a different layer.

$$\begin{aligned} & f\colon V \to \{1,\dots,NL\}\colon \\ & v \to f(v) = i, \\ & \text{where} \quad i \ = \arg\min\nolimits_{\langle v_1,\dots,v_k\rangle\,:\ v_1=u_0, v_k=v} k \end{aligned} \tag{3}$$

The path $\langle v_1, \ldots, v_k \rangle$ is defined here as a sequence of graph vertices $v_1, \ldots, v_k \in V$ such that each vertex is connected with its successor and ancestor $\{v_{i-1}, v_i\} \in E$ $\forall i = 1, \ldots, k$. The path length k is assumed to be minimal (it is the shortest path connecting vertices u_0 and v). Such a path exists for each vertex since the domain is assumed to be compact. NL denotes the resulting number of layers.

Layers and connections. The i-th layer is defined as a subgraph containing all vertices attributed (by the f attributing function) with value i, and all edges between these vertices present in the original graph G.

$$\begin{aligned}
&L_i = \left(V_i^L, E_i^L\right) \subset G\colon \\
&V_i^L \ = \{u \in V\colon f(u) = i\ \} \\
&E_i^L \ = \{\{u, v\} \in E\colon f(u) = f(v) = i\} \\
&i = 1, \ldots, NL
\end{aligned} \tag{4}$$

The i-th connection is defined as a subgraph containing all vertices attributed (by the f attributing function) with i and $i+1$ values, and all edges between vertices assigned to layer i and $i+1$.

$$\begin{aligned}
&I_i = \left(V_i^I, E_i^I\right) \subset G\colon \\
&V_i^I \ = \{u \in V\colon f(u) \in \{i, i+1\}\} \\
&E_i^I \ = \{\{u, v\} \in E, f(u) = i, f(v) = i+1\} \\
&i = 1, \ldots, NL-1
\end{aligned} \tag{5}$$

Graph partition algorithm. Here we describe the *graph-partitioning* algorithm for partitioning the graph G into a set of bipartite subgraphs with disjoint edges. The input for the algorithm is the graph G, and the output is the set of bipartite graphs $\{B_i\}_{i=1,\ldots,NB}$.

As shown in the algorithm description below, in a general case when the graph G is not planar, the total number of bipartite graphs NB is no greater than $h-1$, with

$$\Delta(B_K) \leq h - K + 1, K = 1, \ldots, h-2; \quad \Delta(B_{h-1}) \leq 2 \tag{6}$$

Also, for a planar graph G representing a partitioned $2D$ area, the resulting number of bipartite graphs is not greater than 3, with

$$\Delta(B_1) \leq h; \quad \Delta(B_2) \leq h-1; \quad \Delta(B_3) \leq 2 \tag{7}$$

An example of a planar graph that partitions into 2 bipartite graphs is illustrated in Figure 1. A planar graph with more complex partitioning, requiring sub-layering, is illustrated in Figure 2.

Through the layer-connection algorithm, the task of message scheduling over the entire graph G is reduced to scheduling over independent bipartite graphs, which are simple and can be solved in polynomial time.

The graph partitioning algorithm steps are:

- **Step (1):** Execute the layer-connection algorithm over entire graph G. The algorithm delivers a set of connections $\{I_i\}_{i=1,\ldots,LN-1}$ and disjoint layers $\{L_i\}_{i=1,\ldots,LN}$ sub-graphs.
- **Step (2):** Create the first bipartite graph as the union of all connection subgraphs $B_1 = \bigcup_{k=0,\ldots,LN-1} I_k$.
- **Step (3):** Let iteration index $K = 0$.
- **Step (4): If** the maximum vertex valency in layer subgraphs $\{L_i\}_{i=1,\ldots,LN}$ is less than or equal to 2, **Then Go To Step (9)**
- **Step (5):** Let iteration index $K = K + 1$.
- **Step (6):** Execute recursively the layer-connection algorithm over each layer subgraph $\{L^i\}_{i=1,\ldots,LN}$. The algorithm delivers a set of K sub-connection $\{I_i^{m,K}\}_{i=1,\ldots,LN}^{m=1,\ldots,LN_i^K-1}$ and disjoint K sub-layers $\{L_i^{m,K}\}_{i=1,\ldots,LN}^{m=1,\ldots,LN_i^K}$ subgraphs. LN_i^K is the number of the K-th sub-layer resulting from execution of the algorithm over the L_i-th layer.
- **Step (7):** The $K + 1$ bipartite graph is defined as the union of all K-th sub-connection subgraphs $B_{K+1} = \bigcup_{i=1,\ldots,LN}^{m=0,\ldots,LN_i^K-1} I_i^{m,K}$ resulting from recursive execution of the layer-connection algorithm.
- **Step (8):** Denote by L_i all K-th sub-layer subgraphs, and by I_i all K-th sub-connection subgraphs. Denote LN as the total number of K-th sub-layer subgraphs. **Go To Step (4)**.
- **Step (9):** The $K+2$ bipartite graph is defined as the union of all layer subgraphs $B_{K+2} = \bigcup_{i=1,\ldots,LN} L_i$.
  Stop

The correctness and efficiency analysis of the above algorithm are described in the *Correctness of graph partition algorithm* and *Efficiency of graph partition algorithm* sections.

3. Scheduling over bipartite graphs

We assume that

- Each processor can only send or receive one message at one time.
- Each processor sends different messages to different neighbors.
- The communication cost between each discrete pair of communicating processors is not the same.

The cost of each communication can be bounded as $t_{comm} = \tau + L\sigma$ where τ is the message initialization time, L is the number of words in the message, and σ is the time per byte at which processor can send or receive data through the network. In reality, σ may depend on the number of processor pairs trying to send a message at the same time, and the available routes within the interconnection.

If we assume the **size of each message is the same**, the scheduling problem is equivalent to the problem of coloring of edges of a graph representation of the computational domain, where each vertex denotes a processor and each edge denotes a message to be sent.

A *k-edge-coloring* of graph G is an assignment of k colors to the edges of G in such a way that any two edges meeting at a common vertex are assigned different colors. If G can be k-edge colored, then G is said to be k-edge colorable. The *chromatic index of* G, denoted by $X`(G)$, is the smallest $k$ for which $G$ is $k$-edge-colorable. From *Konig's Theorem* [15] it follows that if $G$ is a bipartite graph whose maximum vertex valency is $d$, then $X`(G) = d$. The idea of the proof is mathematical induction on the number of edge of G. All trees are bipartite graphs [15].

When a graph to be partitioned **is** a representation of a $2D$ mesh, there are not more than three bipartite subgraphs and equation (7) implies that the chromatic index (number of colors required to color the entire graph G) is no greater than $h + h - 1 + 2 = 2h + 1 = O\left(h\right)$. The total communication cost is then bounded by

$$t_{comm} = \left(\tau + L\sigma\right)\left(2h + 1\right)2 = O\left(\tau h + L\sigma h\right) \tag{8}$$

This corresponds to the formulae (1).

More generally, when a graph **is not** a representation of $2D$ mesh, then (6) implies that the entire graph G can be colored by using not more than $\sum_{K=1,h-2}\{h - K + 1\} + 2 = \frac{1}{2}h^2 + \frac{1}{2}h - 3 = O\left(h^2\right)$ colors, in a pessimistic case. The total communication cost is then bounded by

$$t_{comm} = \left(\tau + L\sigma\right)\left(\frac{1}{2}h^2 + \frac{1}{2}h - 3\right)2 = O\left(\tau h^2 + L\sigma h^2\right) \tag{9}$$

The presented upper bound estimates consider a pessimistic case, and usually the algorithm deals better with the input graph. For example, on a regular $2D$ mesh with maximum vertex valency 6, as the one presented in Figure 1, there are only two sets of messages to be sent within 6 time units, and this result does not depend on the number of processors represented by the graph vertices.

In the best known bipartite graph edge coloring algorithms, the complexity is $O\left(n \log n \log h + e\ \log h\right)$; where n is number of vertices, e is number of edges, and h is the maximum vertex valency in the graph [14].

If we assume that the **size of each message is not the same**, the scheduling problem can also be solved by the same strategy above. It is only necessary to select a minimum size message, divide all large messages into a sequence of minimal-sized messages, and replicate the edges in the bipartite graphs representing the largest messages. In the new graph, a single edge between a pair of processors is a minimum size message, and multiple edges constitute a large-sized message.

4. h-Relation personalized communication strategy

In the case when the communication pattern arises from partition of computational mesh and different messages with different sizes must be exchanged between processors representing sub-domains adjacent in a geometrical sense, the proposed communication strategy over a heterogeneous interconnection network can be summarized in the following steps:

1. Create a graph G describing the partitioned mesh as presented in (2). The maximum vertex valency in the graph is denoted by $\Delta(G) = h$.
2. Execute the graph partitioning algorithm described in section *Graph partition algorithm*. The algorithm results in a set of bipartite graphs $\{B_i\}_{i=1,\ldots,NB}$.
3. Schedule messages over each bipartite graph, as described in a section *Scheduling over bipartite graphs*:
   (a) Find the minimum and maximum size of messages represented by edges in the B_i graphs. Split each message larger than the minimum into a sequence of messages with a length equal to the minimum size message (not that the last one will usually contain a shorter message). Replicate each edge by the number of messages in the sequence.
   (b) Color the bipartite graph with multiple edges using an efficient algorithm as the one presented in [14].
   (c) Schedule messages according to the edge coloring.

In the case of a partitioned $2D$ mesh, the graph partition algorithm produces no more than three bipartite graphs, and in the general 3D case there are no more than $h-1$ bipartite graphs, as shown in in the next section.

5. Correctness of the graph partition algorithm

Algorithm Correctness: general case. The analysis in this section applies to a simple geometrical consideration over any graph representation of a $3D$ partitioned domain, where vertices denote sub-domains, and edges denote adjacency (neighbors that share a common border — a non-empty $2D$ area). A *planar graph* is a graph representation of a partitioned $2D$ domain. This section deals with graph representation of partitioned $3D$ meshes, which are not planar graphs.

Lemma 1. *The execution of the layer-connection algorithm over graph G with $\Delta(G) \leq h$ produces a connection subgraph I with $\Delta(I) \leq h$, and a layer sub-graph L with $\Delta(L) \leq h - 1$.*

Proof. Execution of the algorithm results in $\{L_i = (V_i^L, E_i^L)\}_{i=1,\ldots,LN}$ and $\{I_i = (V_i^I, E_i^I)\}_{i=1,\ldots,LN-1}$ subgraphs. Connection subgraph I and layer subgraph L are defined as unions $I = \bigcup_{k=0,\ldots,LN-1} I_k$ and $L = \bigcup_{i=1,\ldots,LN} L_i$.

Each connection subgraph I_k, $k = 1, \ldots, LN-1$ contains all vertices from layers L_k and L_{k+1}, but does not contain edges between vertices within the same layer. If the vertex $v = u_0 \in V_1^L$ has valency h and all neighbors of v are in the next layer, then $\Delta(I_1) = h$. If there are neighbors of the vertex v in graph G within the same layer L_i as v, then $\Delta(I_i) < h$. We conclude that $\Delta(I) \leq h$.

Any vertex $v \in V_i^L$ has valency no greater then $h-1$ in L_i, since:

- If $i=1$, $v \in L_1$, v is the initial vertex $v = u_0$ then v has no edge in L_1, because all of its edges are assigned to the connection subgraph I_1. It simply implies that the valency of $v = u_0$ is 0 in L_1.
- If $i > 1$, then each vertex within L_i has an edge to some vertex from the previous layer L_{i-1}, from definition of layers-connections algorithm. If $v \in L_i$ has h neighbors in G, then at least one of its neighbors belongs to the previous layer L_{i-1}. This implies that v may have a maximum of $h-1$ neighbors in L_i; in other words, the maximum valency of v in L_i is $h-1$.

We conclude that $\Delta(L) \leq h-1$. □

Lemma 2. *Each connection subgraph I_i is a bipartite graph.*

Proof. There are two disjoint sets of vertices in each I_i graph, the first one contains all vertices from layer L_i and the second one contains vertices from layer L_{i+1}. From the definition of the connection subgraph construction there are no edges between vertices inside each set; hence the union of the two layer graphs is a bipartite graph. □

Lemma 3. *The union of all connection subgraphs $I = \bigcup_{k=0,\dots,LN-1} I_k$ is a bipartite graph.*

Proof. Each connection subgraph I_i contains all vertices from layers L_i and L_{i+1}. Vertices from the union graph $I = (V^I, E^I)$ can be split into two disjoint sets $V^I = V_1^I \cup V_2^I$, where V_1^I is the union of all vertices from all odd layers $V_1^I = \bigcup_{k=0,\dots,\lfloor\frac{LN-1}{2}\rfloor} V_{2k+1}^L$ and V_2^I is a union of all vertices from all even layers $V_2^I = \bigcup_{k=1,\dots,\lfloor\frac{LN-1}{2}\rfloor} V_{2k}^L$.

There are no edges between vertices from the same layer. There are also no edges between vertices from layer L_i and L_{i+2} and beyond. This implies that the union graph I is a bipartite graph. □

Theorem 1. *Any graph G is a union of $h-1$ bipartite graphs $\{B_i\}_{i=1,\dots,h-1}$, where $\Delta(B_i) \leq h-i+1, i = 1,\dots,h-2$ and $\Delta(B_{h-1}) \leq 2$.*

Proof. By executing the layer-connection algorithm over G we obtain a connection sub-graph I_1 with $\Delta(I_1) \leq h$ and layer sub-graphs L with $\Delta(L) \leq h-1$, according to 1. By executing the algorithm recursively $h-2$ times over layer sub-graphs, we obtain a set of connection subgraphs $B_k = I_k$, $k = 1,\dots,h-2$ and the layer subgraph $B_{h-1} = L_{h-1}$ with $\Delta(L_{h-1}) \leq 2$. Each connection sub-graph I_k, $k = 1,\dots,h-2$ is a bipartite graph by 3. The terminal layer subgraph, B_{h-1}, has a maximum vertex valency 2, which implies that it is also a bipartite graph. The procedure is illustrated in Figure 3. □

Theorem 1 establishes the correctness of the layer-connection algorithm in the general case.

Algorithm correctness: graph representations of $2D$ meshes. The analysis in this section applies to a simple geometrical consideration over *planar* graphs. Graph representations of partitioned $2D$ meshes are planar graphs.

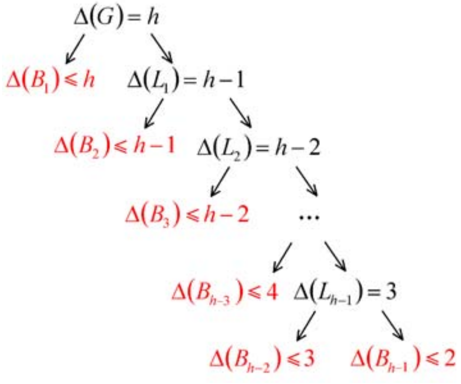
$\Delta(G)=h$
$\Delta(B_1)\leqslant h$
$\Delta(L_1)=h-1$
$\Delta(B_2)\leqslant h-1$
$\Delta(L_2)=h-2$
$\Delta(B_3)\leqslant h-2$
...
$\Delta(B_{h-3})\leqslant 4$
$\Delta(L_{h-1})=3$
$\Delta(B_{h-2})\leqslant 3$
$\Delta(B_{h-1})\leqslant 2$

Fig. 3. Partition of any graph G with maximum vertex valency h into $h-1$ bipartite graphs

For clarity, we assume that the domain is convex without holes. However, the analysis can be easily extended to the case of non-convex domains with holes. We make the following observations:

- All layers are created by passing of a wave from an initial sub-domain vertex u_0, selected randomly, located on the boundary of the domain.
- Each layer represents a sequence of connected sub-domains. The first and the last sub-domains are located on the boundary of the domain.
- Within a layer, there exists paths between each pair of vertices. Each edge represents adjacency between sub-domains.
- We define an *internal path* as one that passes through the entire layer L_i, and we denote edges within this path as *internal edges*. An internal path contains all vertices in the layer. We order all vertices representing sub-domains in L_i from "top" to "bottom", where by the "upper" sub-domain we mean the one located on the upper boundary of the domain. Figure 4 illustrates the ordering.

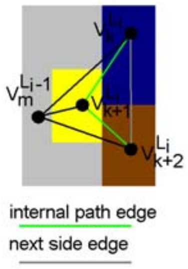

Fig. 4. Example of internal path and next layer edges

We enumerate vertices according to the order and denote them by $< v_1^{L_i}, \ldots, v_p^{L_i} >$, where p is the length of the path.

- In general, all non-internal path edges may be the result of adjacency on the side of the previous layer, or on the side of the next layer, since edges represent geometrical $2D$ connections between sub-domains.
- We distinguish two sides of a layer for assigning a “side” to connections between sub-domains (that are not in the internal path defined above): the *previous layer side* and the *next layer side.*
- Within a layer, there are three kinds of edges: *internal edges*, *previous side edges* and *next side edges*, according to their geometrical location, as illustrated in Figs. 4, 5.

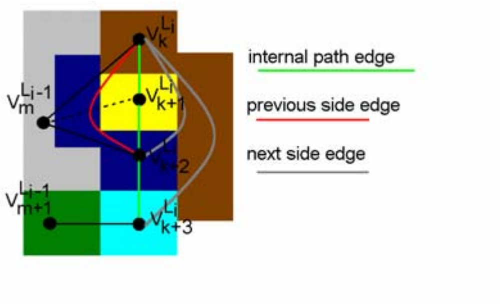

Fig. 5. Covering of the sub-domain $v_{k+1}^{L_i}$ by the previous side edge connections

Lemma 4. *There are no previous side edges.*

Proof. Let us suppose there is a previous side edge $\{v_k^{L_i}, v_{k+l}^{L_i}\}$ with $l \geq 2$, see Figure 5. From the definition of the layer construction, each vertex $v_k^{L_i}$ must be connected with some vertex from the previous layer. Existence of a previous side edge $\{v_k^{L_i}, v_{k+l}^{L_i}\}$ means that the sub-domain represented by vertex $v_k^{L_i}$ is adjacent to the sub-domain represented by vertex $v_{k+l}^{L_i}$ through a common vertex in the previous layer side. But previous-side adjacency of sub-domain $v_k^{L_i}$ to sub-domain $v_{k+l}^{L_i}$ for $l \geq 2$ means that sub-domains $v_{k+1}^{L_i}, \ldots, v_{k+l-1}^{L_i}$ are “covered” from any possible connection to the previous layer. Hence, the existence of previous-side edges is a contradiction with the fact that the covered vertices are adjacent to some vertex $v_m^{L_{i-1}}$ in the previous layer. □

Lemma 5. *Next side edges do not intersect themselves.*

Proof. Let us suppose there is a next side edge $\{v_k^{L_i}, v_{k+m}^{L_i}\}$ intersecting with a next side edge $\{v_{k+l}^{L_i}, v_{k+n}^{L_i}\}$ where $m > 2, n - l > 2$ and $1 < l < m < n$. Adjacency of sub-domain $v_k^{L_i}$ to sub-domain v_{k+m} means that sub-domains $v_{k+1}^{L_i}, ...v_{k+l}^{L_i}, \ldots, v_{k+m-1}^{L_i}$ are covered on the next layer side by the $\{v_k^{L_i}, v_{k+l}^{L_i}\}$ connection. Hence, this contradicts the fact that sub-domain $v_{k+l}^{L_i}$ can also be connected to sub-domain $v_{k+n}^{L_i}$ on the next side. □

Lemma 6. *The second execution of the layer-connection algorithm over all layers L_i with $\Delta(L_i) = h-1$ results in one bipartite graph SI with $\Delta(SI) \leq h-1$ and one graph SL with $\Delta(SL) \leq 2$.*

Proof. By executing the layer-connection algorithm over each layer L_i, we obtain a set of bipartite *sub-layers* subgraphs $\{L_i^m\}_{i=1,\ldots,LN-1}^{m=1,\ldots,LN_i-1}$ and a set of *sub-connection* subgraphs $\{I_i^m\}_{i=1,\ldots,LN-1}^{m=1,\ldots,LN_i-1}$, where LN_i denotes the number of sub-layers resulting from execution of the algorithm over the layer L_i.

The union of all sub-connection subgraphs is a bipartite subgraph, $SI = \bigcup_{i=1,\ldots,LN-1}^{m=1,\ldots,LN_i-1} L_i^m$, by 3. Also, $\Delta(SI) \leq h-1$, since $\Delta(L_i^m) \leq h-1$, $i = 1,\ldots,LN$, for each layer according to 1.

It is necessary to show that $\Delta(SL) \leq 2$, where SL is the union of sub-layers L_i^m resulting from the execution of the algorithm over all layers L_i. We can do this by proving the same property for each sub-layer, $\Delta(L_i^m) \leq 2$, $i = 1,\ldots,LN$, $m = 1,\ldots,LN_i$, since all sub-layers are disjoint. In other words, it is only necessary to show that each vertex $v_k^{L_i^m}$ in any sub-layer L_i^m has a valency no greater then 2 in L_i^m.

We proceed with the proof by considering all adjacency cases of the vertex $v_k^{L_i^m}$, and observations about the cases (each sub-layer has all the properties of a layer):

- The internal path in the sub-layer is an ordered set of vertices along the path $< v_1^{L_i^m}, \ldots, v_p^{L_i^m} >$.
- We can assume that the initial vertex $v_1^{L_i^1}$ is the first vertex in the internal path, and is located on the boundary of the domain.
- Each vertex $v_k^{L_i^m}$ from sub-layer L_i^m can only have two kind of edges: *internal edges* and *next side edges*, since *previous side edges* are not possible, according to 4.
- Each vertex $v_k^{L_i^m}$ in sub-layer L_i^m for $m > 1$ must have an edge to some vertex $v_l^{L_i^{m-1}}$ from the previous sub-layer L_i^{m-1}.
- Each vertex $v_k^{L_i^m}$ in sub-layer L_i^m can have neighbors in the layer L_i only from layers L_i^{m-1} or L_i^m or L_i^{m+1}, from the properties of the layer-connection algorithm.
- We distinguish *forward* and *backward* directions along the internal path according to the order of vertices on the internal path.

First, we consider the following **observations** for the possible adjacency cases in the backward direction:

1. Each vertex $v_k^{L_i^m}$ from sub-layer L_i^m can only have one internal edge associated with the vertex $v_{k-1}^{L_i^m}$ and many next side edges.
2. If there is a next side edge $\{v_l^{L_i^m}, v_k^{L_i^m}\}$ between two vertices in sub-layer L_i^m, then between those vertices there can only be vertices from further sub-layers

$L_i^{m+n}, n \geq 1$. Assume between them there is a vertex $v_{k+n}^{L_i^m}, n \in \{1, \ldots, l-1\}$ from the same sub-layer, see Figure 6(a). The vertex $v_{k+n}^{L_i^m}$ must be connected with some vertex from the previous sub-layer $v^{L_i^{m-1}}$, by the definition of sub-layer. There is no possibility of connecting internal vertices $v_{k+1}, \ldots, v_{l-1}$ with any external vertex, because all the vertices are covered by the next side edge, and edges do not intersect.

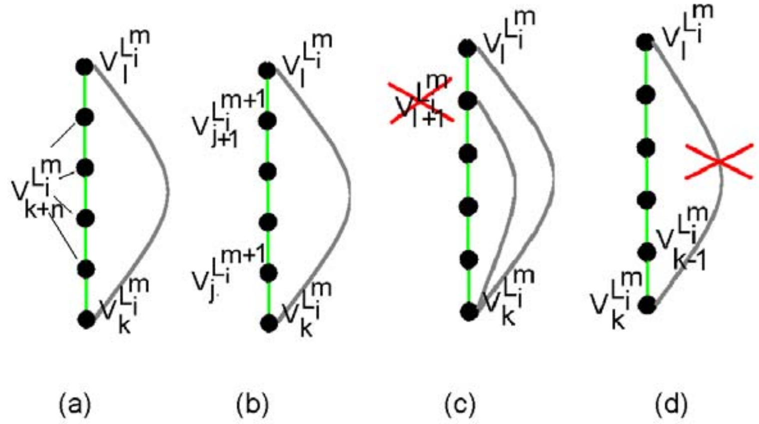

Fig. 6. Adjacency cases for vertex from sub-layer: **(a)** Vertices covered by the edge $\{v_k^{L_i^m}, v_l^{L_i^m}\}$. **(b)** Internal edges are from the next sub-layer. **(c)** Only one edge from the same sub-layer through the next sub-layer edge. **(d)** Only one neighbor from the same sub--layer through the internal edge

3. If the vertex $v_k^{L_i^m}$ from sub-layer L_i^m is connected by a next side edge with another vertex $v_l^{L_i^m}$ from the same layer in the backward direction, then the neighbor $v_j^{L_i^{m+1}}$ of the vertex $v_k^{L_i^m}$ through the internal edge in the backward direction is from the next layer L_i^{m+1}, see Figure 6(b). Let us consider the execution of of the layer-connection algorithm over the layer L_i. The m-step assigns vertices $v_k^{L_i^m}$ and $v_l^{L_i^m}$ to sub-layer L_i^m. In the next step, v_j will be assigned into a next sub-layer L_i^{m+1}.
4. Each vertex $v_k^{L_i^m}$ from sub-layer L_i^m can only have one neighbor $v_l^{L_i^m}$ through the next side edge from the same sub-layer in the backward direction. Suppose it has two such neighbors $v_l^{L_i^m}$ and $v_{l+1}^{L_i^m}$, see Figure 6(c). Then, the neighbor $v_{l+1}^{L_i^m}$ from sub-layer L_i^m is covered by the edge $\{v_k^{L_i^m}, v_l^{L_i^m}\}$, which is a contradiction with **observation 2**.

5. Each vertex $v_k^{L_i^m}$ from sub-layer L_i^m can only have one neighbor within the same sub-layer in the backward direction. The reason is the following: The vertex $v_k^{L_i^m}$ cannot have two neighbors from the same sub-layer through the next side edges in the backward direction, according to **observation 4**, see Figure 6(c). If the vertex has only one neighbor from the same sub-layer through next side edge in the backward direction, then its internal edge neighbor is not from the same sub-layer, from **observation 3**, see Figure 6(b). If the vertex $v_k^{L_i^m}$ has an internal path neighbor from the same sub-layer in the backward direction, then the vertex $v_k^{L_i^m}$ cannot have another neighbor through a next sub-layer edge, compare Figure 6(d).

We conclude that the vertex $v_k^{L_i^m}$ may only have one neighbor $v_{k-1}^{L_i^m}$ from the same sub-layer when traversing the internal path in the backward direction. The same observations apply in the forward direction, since the vertex $v_k^{L_i^m}$ from sub-layer L_i^m can only have two neighbors $v_{k-1}^{L_i^m}$ and $v_{k+1}^{L_i^m}$ in the same sub-layer as determined from a scan in both directions, $\Delta L_i^m \leq 2$. □

Theorem 2. *If the graph G is planar and $\Delta(G) = h$, then G is a union of no more than 3 bipartite graphs; that is, $G = \bigcup_{i=1,\dots,3} B_i$, with $\Delta(B_1) \leq h$, $\Delta(B_2) \leq h - 1$ and $\Delta(B_3) \leq 2$.*

Proof. The theorem holds as a consequence of Lemma 1 and Lemma 6. The recursive partitioning procedure in Figure 3 is truncated at the second recursion level as illustrated in Figure 7, because the sub-graph layers have a maximum vertex valency of 2. □

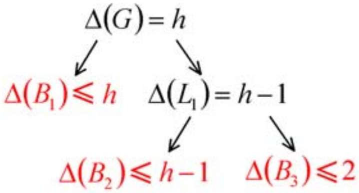
$\Delta(G) = h$
$\Delta(B_1) \leqslant h$
$\Delta(L_1) = h - 1$
$\Delta(B_2) \leqslant h - 1$
$\Delta(B_3) \leqslant 2$

Fig. 7. Partition of planar graph G with maximum vertex valency h into 3 bipartite graphs

The theorem proves the implied correctness of the layer-connection algorithm in the case of planar graph representing a partition of a $2D$ mesh.

6. Complexity of the graph partition algorithm

Algorithm complexity for the general case. In the general case the graph partition algorithm produces $h - 1$ bipartite graphs, where h is the maximum vertex valency in the graph.

We assume that the graph is stored as a data structure where graph vertices and edges are kept in a variable-length list. The operation of accessing all edges of all vertices in the list has a complexity $O\,(nh)$, where n denotes the number of vertices in the graph. The complexity of each step (See **Graph partition algorithm**.) in the algorithm is described below:

1. **Step(1)** is $O\,(nh)$. The layer-connection algorithm visits each vertex, browses all of its neighbors, and assigns layer-connection indexes to each vertex.
2. **Step(2)** is $O\,(nh)$. This step browses all vertices and edges of the graph, to form a bipartite graph B_1 and layer subgraph L_1. It is performed by attributing graph edges and vertices.
3. **Step(3)** is $O\,(1)$. Initializes a layer counter.
4. **Step(4)** is $O\,(nh)$. This step searches all vertices of the graph, and all edges of particular layers, to find the maximum valency of the graph vertices. All edges of all graph vertices are browsed, to determine their assignment to layers. layer.
5. **Step(5)** is $O\,(1)$. The layer counter is incremented.
6. **Step(6)** is $O\,(nh)$. This step required browsing of all vertices of the graph and all edges of those vertices assigned to particular layers. All edges of the graph vertices are checked to determine their layer assignment.
7. **Step(7)** is $O\,(nh)$. This step queries all vertices of the graph, and all edges of particular layers, to group them into a bipartite graph B_{k+1} and the sub-layers of subgraph L_{k+1}. This is accomplished by attributing graph edges and vertices. All edges of all graph vertices are searched for assignment to a sub-layer.
8. The **Step(8)** This is a pseudo-code description for defining a naming convention.
9. **Step(9)** is $O\,(nh)$. This step requires browsing all vertices of the graph, and all edges of particular sub-layers, for attributing them to the last bipartite graph B_{k+2}. All graph edges of the graph vertices are searched for assignment to a sub-layer.

In the general case, **Step(4)**–**Step(8)** are repeated $h-2$ times. The total computational complexity contribution from the above list is

$$O\,(nh + nh + 1 + (h-2)\,(nh + 1 + nh + nh) + nh) = O(nh^2) \qquad (10)$$

Algorithm complexity for a planar graph. The case of planar graph differs from the general case only in the number of iterations. The **Step(4)**–**Step(7)** loop is performed no more than one time. Hence, the total computational complexity of the algorithm for a planar graph is

$$O\,(nh + nh + 1 + nh + 1 + nh + nh + nh) = O(nh) \qquad (11)$$

7. Conclusions

- We have shown that all communication problems arising from a partition of a 2D computational domain into sub-domains and exchanging data with neighbors can be solved using a h-relation personalized communication (h-*rpc*), where h is the maximum number of neighboring sub-domains.

- We propose an alternative *h-rpc* strategy for communication patterns arising from partition of the domain into sub-domains. The strategy uses an algorithm that partitions a graph representation of the domain into a set of bipartite graphs, edge-colors the bipartite graphs, and schedules communication according to the assigned color.
- We have shown the correctness of proposed graph partitioning algorithm and analyzed its computational complexity.
- We have derived the cost for scheduling our bipartite graphs edge-coloring algorithm.
- For planar graphs representing partitioned $2D$ areas, the number of resulting bipartite graphs is not more that 3, and the number of colors required for edge-coloring is not greater than $2h + 1$.
- For the graph representation of partitioned $3D$ areas, the number of resulting bipartite graphs is not more that $h - 1$, and the number of colors required for edge-coloring is no greater than $\frac{1}{2}(h^2 + h - 6)$.

Acknowledgements

The work has been partially suported by Polish MNiSW grant no. NN 519 447739.

References

[1] Bader D. A.: *High-Performance Algorithms and Applications for SMP Clusters.* NASA High Performance Computing and Communications Aerospace Workshop CAS 2000, Moffett Field, CA, 2000.

[2] Bader D. A., Helman D. R., JáJá J.: *Practical Parallel Algorithms for Personalized Communication and Integer Sorting.* ACM Journal of Experimental Algorithmics, vol. 1, 1996, pp. 1–42.

[3] Biggs N. L.: *Discrete Mathematics.* Oxford University Press, 1985.

[4] Cormen T. H., Leiserson C. E., Rivest R. L.: *Introduction to Algorithms.* MIT Press, 1999.

[5] Goldman A., Trystram D.: *Algorithms for the Message Exchange Problem.* Proc. of the International Conference on Parallel Computing in Electrical Engineering, Poland, 1998.

[6] Jain R., Werth J., Browne J.C., Sasaki G.: *A graph-theoretic model for scheduling problem and its application to simultaneous resource scheduling.* Computer Science and Operations Research: New Developments in Their Interfaces, Penguin Press, 1992.

[7] Johnsson S. L., Ho. C.-T.: *Optimal All-to-All Personalized Communication with Minimum Span on Boolean Cubes.* Technical Report 18–91, Harvard University, 1991.

[8] Johnsson S. L., Ho. C.-T.: *Optimal Broadcasting and Personalized Communication in Hypercubes.* IEEE Transactions on Computers, vol. 38, 1989, pp. 1249–1268.

[9] Kapoor A., Rizzi R.: *Edge Coloring Bipartite Graphs.* Technical Report DIT-02-040, University of Trento, 1999.

[10] Marathe M. V., Panconesi A., Risinger L. D.: *An experimental study of a simple, distributed edge coloring algorithm.* Proc. of the 12th annual ACM symposium on Parallel Algorithms and Architectures, Bar Harbor, Maine, 2000, pp. 166–175.

[11] Mathur K. K., Johnsson S. L.: *All-to-All Communication Algorithms for Distributed BLAS* Technical Report 07-93, Harvard University, 1993.

[12] Mathur K. K., Johnsson S. L.: *Communication Primitives for Unstructured Finite Element Simulations on Data Parallel Architectures.* Technical Report 23-92, Harvard University, 1992.

[13] Risinger L. D., Marathe M. V., Panconesi A.: *Edge Coloring Algorithms for Scheduling in ASCI: Survey and Experimental Results.* Technical Report LAUR-No-97-2341, Los Alamos National Laboratory, 1997.

[14] Rizzi R.: *Finding* 1*-Factors in Bipartite Regular Graphs and Edge-Coloring Bipartite Graphs.* SIAM J. Discrete Math., vol. 15, 2002, pp. 283–288.

[15] Skiena S.: *Coloring Bipartite Graphs.* [in:] Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, Addison-Wesley, 1990.

[16] Sundar N. S., Jayasimha D. N., Panda D. K., Sadayappan P.: *Hybrid Algorithms for Complete Exchange in 2D Meshes.* IEEE Transactions on Parallel and Distributed Systems, vol. 12, 2001, pp. 1201–1218.

[17] Wang J.-C., Lin T.-H., Ranka S.: *Distributed Scheduling of Unstructured Collective Communication on the CM-5.* Parallel Processing Letters, special issue on Partitioning and Scheduling, 1995.

[18] Wenjie H., Xiaoling H., Ko-Wei L., Jiating S., Weifan W., Xuding Z.: *Edge-Partitions of Planar Graphs and Their Game Coloring Numbers.* Journal of Graph Theory, vol. 41, 2002, pp. 307–317.

[19] Zhi-Zhong C., Xin H. *Parallel complexity of partitioning a planar graph into vertex-induced forests.* Discrete Applied Mathematics, vol. 69, 1996, pp. 183–198.