J Math Chem (2009) 45:503–512
DOI 10.1007/s10910-008-9423-2

ORIGINAL PAPER

Communication performance of *d*-meshes in molecular dynamics simulation

Roman Trobec · Urban Borštnik · Dušanka Janežič

Published online: 8 August 2008

Abstract Communication algorithms, tailored for molecular dynamics simulation on d-meshes, are evaluated in terms of communication efficiency. It has been shown elsewhere that d-meshes are better than other regular topologies, e.g., hypercubes and standard toroidal *4*-meshes, when compared in their diameter and average distance among nodes. Collective communication is needed in molecular dynamics simulation for the distribution of coordinates and calculation and distribution of new energies. We show that both collective communication patterns used in molecular dynamics can be efficiently solved with congestion-free algorithms for all-to-all communication based on store-and-forward routing and routing tables. Our results indicate that d-meshes compete with hypercubes in parallel computers. Therefore d-meshes can also be used as a communication upgrade of existing molecular dynamics simulation platforms and can be successfully applied to perform fast molecular dynamics simulation.

Keywords d-meshes · Parallel molecular dynamics simulation · Computer interconnection topologies

1 Introduction

Large-scale natural phenomena, experiments that would cost vast amounts of money, those that are ecologically problematic or dangerous, or those that can not be implemented experimentally [1], can first be simulated in order to predict the outcome

R. Trobec
Jožef Stefan Institut, Jamova 39, Ljubljana, Slovenia

U. Borštnik · D. Janežič (✉)
National Institute of Chemistry, Hajdrihova 19, Ljubljana, Slovenia
e-mail: dusa@cmm.ki.si

[2–4]. Such simulations need a significant amount of computing. Parallel computer clusters provide the computational rates necessary for mentioned computer simulations.

Currently the more slowly increasing rate in processor performance is being temporarily compensated by larger numbers of parallel processors that cooperate on the same problem. Today, it is not unusual to utilize parallel machines with several thousand processors; however such huge parallel machines are not always able to execute numerical algorithms efficiently. It is known that only scalable algorithms can be implemented efficiently on a massive parallel machine [5]. Typically, such algorithms need only a small amount of communication between processors. By increasing the number of processors, the computational load on each processor decreases, but the communication requirements increase. There is a limit beyond which any further increase in the number of processors is not profitable since the computational load becomes too small compared to the communication load. Therefore, beside the increased computational performance, the communication speed should also be increased.

The efficiency of any parallel algorithms is to some degree limited by their communication patterns on a chosen topology [6]. For example, in molecular dynamics simulation, all-to-all collective communication patterns must be performed for the distribution of updated atomic coordinates and the calculation of new forces for all particles. The amount of communication increases with the number of processors [6,7] and in this way limits the speed-up of the simulation. We will focus in this paper on the communication speed, which can be increased with an addition of extra links to an existing interconnection network in a computational cluster. It is assumed that the routing switch is a separate component of the computing cluster, which is able to manage all communication traffic. Calculation and communication can be performed in parallel to some extent. Routing is implemented by routing tables and separate communication queues on each communication channel. Processing units are thus loaded only by their own communicated data [8]. Such architectures can provide a moderate number of communication links on each processing node. We further suppose an isomorphic interconnection network, composed of the same architecture of network nodes, even if the number of nodes is increased.

Communication patterns of established topologies, such as meshes and hypercubes, have been investigated in detail [9]. Meshes, perhaps the most widely used today, have well-known preferences and weaknesses [10,11]. Their preferences are primarily in the scalability and simple routing algorithms [12]. The simplicity of these algorithms is due to their simple structure. One of the weaknesses of meshes is their relatively large diameter, which prevents the efficient broadcast and exchange of messages among all system nodes. With the introduction of additional links as proposed in d-meshes [13], the weakness of a small diameter is mitigated. In order to take advantage of additional links, new routing algorithms must be applied. In this paper, d-meshes are analyzed regarding the communication patterns that are needed for parallel molecular dynamic simulation. A new congestion-free algorithm for all-to-all routing is proposed and analyzed, restricted to message-passing based on the store-and-forward method [9].

The rest of the paper is organized as follows. First, a short background of molecular dynamics simulation is given with an analysis of the required communication patterns. In Sect. 3, d-meshes are described and the new all-to-all routing algorithm is explained. Finally, the comparative performance results are given for different topologies. The paper concludes with some directions for future work.

2 Communication patterns in parallel molecular dynamics

Molecular dynamics simulation is based on the solution of a system of Newtonian equations of motion for N atoms [14]. The solution, evolving in time, is obtained by a simple numerical integration in a sequence of time steps, each of which includes an energy calculation and a coordinate update. In parallel computation, calculations normally performed on a single processor are distributed among P processors of the parallel computer. In the parallel molecular dynamics, both the energy calculation and the coordinate updates are parallelized. Each processor calculates a fraction, $1/P$ of the calculations, needed to calculate total energy, and performs N/P of the N atoms coordinate updates. Because of the sequential nature of the simulation time steps, the energy calculation and the coordinate updates must follow sequentially: the energy must be known to update the coordinates, and the updated coordinates must be known to calculate the new energy [14].

In parallel molecular dynamics simulation, processors communicate data between the energy calculation and the coordinate updates [15]. After the calculation of energy, each processor has only $1/P$ of the total system energy. The sum of these partial energies is equal to the total energy, which is the same as what is calculated by a single processor. At each time step, all processors perform a collective reduce-and-scatter communication. The collective reduction produces the total sum of the energy, while the scatter operation distributes the calculated total energy among the processors, because each processor needs only the new energy of their N/P atoms for which it will update the coordinates [16]. After the coordinate update, all of the processors broadcast the new coordinates of their local atoms to all other processors using all-to-all broadcast communications. It follows that the processors exchange packets of data in multiples of N/P atoms.

In some traditional approaches to the parallel molecular dynamics, every processor needs the coordinates of all of the atoms to perform the parallel energy calculation [17, 18]. A collective gather communication performs the required all-to-all assembling of new local coordinates, after which every processor retains the updated coordinates for all N atoms and calculates the new energy and coordinates. The processors again exchange packets of data in multiples of N/P atoms. The collective reduce-and-scatter and the collective gather operations are related in the sense that they need all-to-all communication. They differ in the sequence of transfers, which is reversed, and in the additional summation of the received data (partial energies) by the collective reduce-and-scatter operation. It follows that the most time consuming communication of every parallel molecular dynamics algorithm is an all-to-all exchange of messages of length $K \times N/P$, where K is a constant. All-to-all communication patterns will be analyzed in detail in the following sections.

3 Routing in d-meshes

Let the topology of a d-mesh consist of $N = X \times Y$ nodes. The nodes are placed in Y rows each of length X and they can thus be uniquely represented by pairs (x, y), $0 \leq x < X$, $0 \leq y < Y$ with each node (x, y) located in the x^{th} column and y^{th} row. Also, each node has d bidirectional links (where d is an even integer and $d \geq 4$) that can be enumerated as $l_i = (l_{i_x}, l_{i_y})$, $1 \leq i \leq d$, $0 \leq l_{i_x} \leq X$ and $0 \leq l_{i_y} \leq Y$. Let $x \oplus_n y$ be defined as $x +_{(modulo\ n)} y$ and $x \ominus_n y$ as $x -_{(modulo\ n)} y$. The first four links, which must always be present, are $l_1 = (1, 0)$, $l_2 = (-1, 0)$, $l_3 = (0, 1)$, and $l_4 = (0, -1)$. Therefore, in d-meshes each node (x, y) is always connected to nodes $(x \oplus_X 1, y)$, $(x \ominus_X 1, y)$, $(x, y \oplus_Y 1)$, and $(x, y \ominus_Y 1)$, which is also the definition of standard meshes. Consequently, 4-meshes and standard meshes are identical. Another restriction is the requirement of symmetrical links. That is, if the link $l_i = (l_x, l_y)$ is chosen, then the link $l_{i+1} = (-l_x, -l_y)$ must also be used. Therefore d-meshes are completely defined by choosing $d/2 - 2$ links. These links are regarded as free links. Let us emphasize that each node has the same set of links so that the graph remains isomorphic. As an illustration consider an 8-mesh with $N = 256 = 16 \times 16$ nodes and two free links, namely $(4, -1)$ and $(1, -5)$, as shown in Figure 1. Other configurations of free links might be chosen. Having the same number of nodes and links, these selections result in different diameters of corresponding topologies. d-meshes with the smallest possible diameter for a constant N are defined as optimal

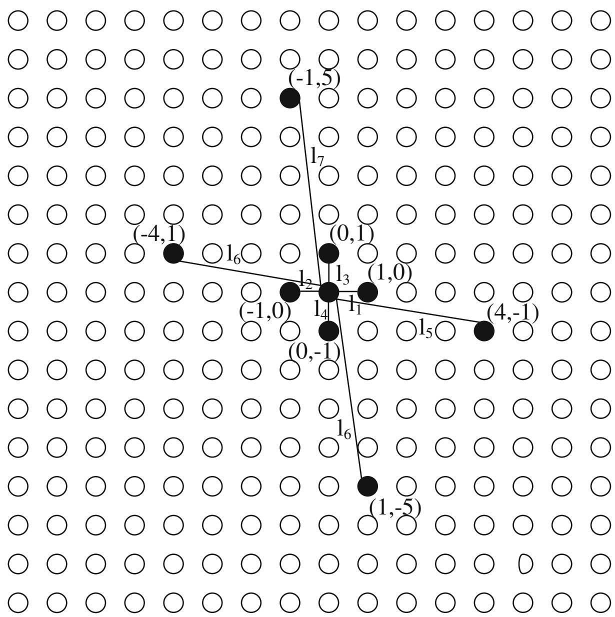

Fig. 1 An optimal 8-mesh for $N = 16 \times 16 = 256$ nodes with the diameter equal to 5. Only directly connected links $(1, 0)$, $(-1, 0)$, $(0,1)$, $(0, -1)$, $(4, -1)$, $(-4, 1)$, $(1, -5)$, $(-1, 5)$ to the central node $(0, 0)$ are shown

d-meshes. The 8-mesh shown in Figure 1 is one of the optimal 8-meshes with $N = 256$ nodes.

The distance between nodes u and v is considered as the number of communication steps (hops) on a shortest path between u and v. Additional introduced free links contribute to a smaller distance and also to a smaller diameter of the topology. The diameter of the 8-mesh given above is 5. The diameter of a 4-mesh with the same number of nodes is 16, while the diameter of a hypercube with 256 nodes is 8. Therefore, the optimal 8-mesh is better than the hypercube of the same size and degree regarding the diameter. A comparison of d-meshes with other topologies in terms of their diameters and average distances can be found in [13]. However, the diameter of a topology alone does not suffice for its evaluation and comparison. The routing algorithms for collective communication must also be adapted to efficiently use the additional links.

The current trend in personal computer clusters is the use of multiprocessor computers within a cluster. Such computers are mapped in a straightforward fashion to d-meshes. The processors (most often 2 or 4) are mapped to sequential nodes in the mesh, either vertically or horizontally; e.g., both nodes (0, 0) and (0, 1) would be in the same computer in a 2-processor SMP computer, and similarly for nodes (0, 2) and (0, 3) and further. The number of computers, each with p processors, required for a N-node d-mesh is N/p. It is important to note that all of the processors in one computer share their outside links. If 2-processor computers are arranged in columns as nodes (0, 0) and (0, 1), they share the physical link (1, 0) for the logical links to node (1, 0) and node (1, 1). It must be noted that the link between the processors (0, 0) and (0, 1), implemented by the system bus, is much faster than the outside communication links.

3.1 One-to-one routing

In one-to-one routing, a message is sent from source node $v_{source} = (x_s, y_s)$ to destination node $v_{dest} = (x_v, y_v)$. Minimal paths between nodes can be determined by a computer program that generates routing tables. Using well known approaches such minimal spanning trees (MST) [19] the distances from a node, say (0,0), to any other node, are calculated and stored in a two-dimensional array. The distance from (0, 0) to (x, y) is stored in the cell [x][y] of the array. By using regularity and toroidal properties of d-meshes the distance between v_{source} and v_{dest} is the same as the distance between (0, 0) and $(x_d \ominus_X x_s, y_d \ominus_Y y_s)$. The distances between nodes can therefore be calculated only once.

The minimal path between two nodes is determined by using the distance table. Suppose that the distance between v_{source} and v_{dest} is D. First determine all d neighbors of v_{source} with the distance $D - 1$ to v_{dest}. Sending the message to any of these nodes sets the message one step closer to its destination. Iterating this D times the message arrives to its destination by using some minimal path. After performing the same procedure for each node, the routing table is finished. It size, for every node, is $d \times N$ bits. During the routing through the minimal path, the node router has to extract the destination address (x, y) and read the next hop direction from the routing table. Detailed algorithms for routing in d-meshes are given in [20].

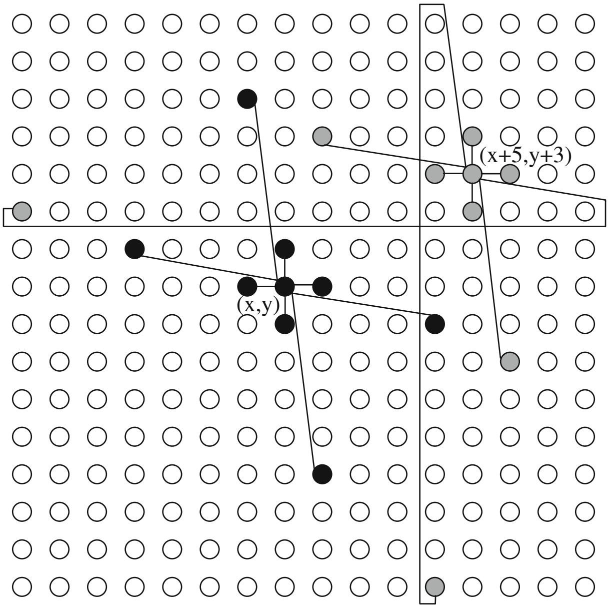

Fig. 2 An example of the translation of a spanning tree of depth 1 in an *8*-mesh, from position(x, y)— black nodes, to position $(x + 5, y + 3)$— gray nodes

3.2 One-to-all routing

For one-to-all communication, routing tables of size $d * N$ bits are needed, in general, on every node. The tables are filled by applying the MST initiated from each node. Information on links belonging to the MSTs is saved in the routing table at the address $addr = y_i X + x_i$. Using MSTs guarantees that the number of steps in one-to-all communication is equal to the diameter of the interconnection network. The expensive multiple calculations can be avoided because of the isomorphism of d-meshes. Because all MSTs are isomorphic, they can be translated along the d-mesh. The translation for (dx, dy) is defined as the movement from (x_i, y_i) to $(x_i \oplus_X dx, y_i \oplus_Y dy)$. If a connection exists in the MST between nodes (x, y) and (w, z), there is also a connection between $(x \oplus_X dx, y \oplus_Y dy)$ and $(w \oplus_X dx, z \oplus_Y dy)$ after translation. An illustrative example of translation is shown in Figure 2. After an initial generation of MST and all possible N translations, the routing tables of all nodes are filled. We could see that the routing tables for one-to-one and one-to-all communication are different; therefore, a communication type flag should be inserted in message headers.

3.3 All-to-all routing

Considering that d-meshes use *4*-meshes as its basis, all-to-all routing algorithm for toroidal *4*-meshes can also be used in d-meshes but the advantage of additional links is then lost. Let us now first consider the algorithm in *4*-meshes. Its idea is based on

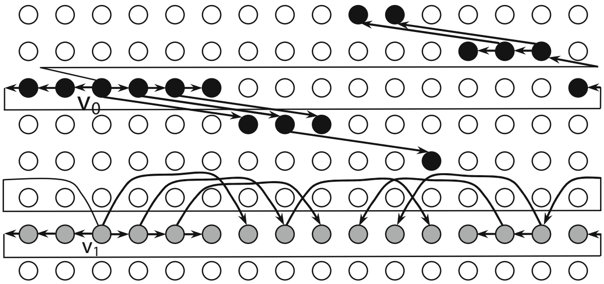

Fig. 3 Example of two of many possible virtual rows for the initiating node v_0 (black nodes) with a free link (4, −1) and v_1 (gray nodes) with free link (4, 0), both with diameter of 3

exchanging the messages among the rows first, to merge these messages into one grouped message and then to exchange these grouped messages among the columns [9,12]. Note that some additional routing logic for merging the messages, switching between rows/columns, and checking for the presence of all messages from rows/columns is needed. The number of required steps for such all-to-all communication in a *4*-mesh is$X/2 + Y/2$.

In d-meshes, additional free links can be used for the distribution of data among a group of connected nodes that constitute a *virtual row*, which contain exactly one node from each column. Each node has its own virtual row and all of them are isomorphic. The same principle is used also for *virtual columns*, with only one node from each virtual row. Consider a simple example (see Figure 3) based on a 6-mesh. The free link of the upper virtual row is $l = (l_x, l_y) = (4, -1)$ and the starting node v_0. In the first step, nodes (1, 0), $(0 \ominus_X 1, 0) = (-1, 0)$, $(0 \oplus_X l_x, 0 \oplus_Y l_y) = (4, -1)$ and $(0 \ominus_X l_x, 0 \ominus_Y l_y) = (-4, 1)$ are added to the group, then (2, 0), $(0 \ominus_X 2, 0) = (-2, 0)$, $(0 \oplus_X l_x \oplus_X 1, 0 \oplus_Y l_y) = (5, -1)$ and $(0 \ominus_X l_x \ominus_X 1, 0 \ominus_Y l_y) = (-5, 1)$ and so on. Before accepting a new node in the group, a check is performed whether the new node is the first from its column. Suppose that we reached the node $(l_x - 1, 0) = (3, 0)$. A new node can not be added by using link (1, 0) because a node from this column is already in the group. The procedure continues with node $(l_x + 2, l_y) = (6, -1)$ and is repeated until all of the column nodes are in the group.

The principle used in the given example can be easily generalized to $2k$ links used for building virtual rows ($2k \leq d - 2$ since at least two links must be used for virtual columns). Due to the symmetry, only the right half of the group is examined. Starting with v_0 in the first step, the nodes connected by k links are added to the group. Then a move to the right node again contributes k new nodes, etc., until an already visited column is reached. Then the nearest group node that can reach new nodes by using link (0,1) is found and the algorithm continues. This is repeated until all the columns are visited. Obviously, the number of steps needed for the formation of a group depends on the free links. The most efficient links would divide the right part of a group with $X/2$ nodes on k equidistant intervals. The length of interval determines the number of steps needed for broadcasting the message to nodes in the right part of the group, and, due to the symmetry, also for the left part, which results in X/k steps. The links needed for construction of a group are stored in all-to-all routing tables. In general,

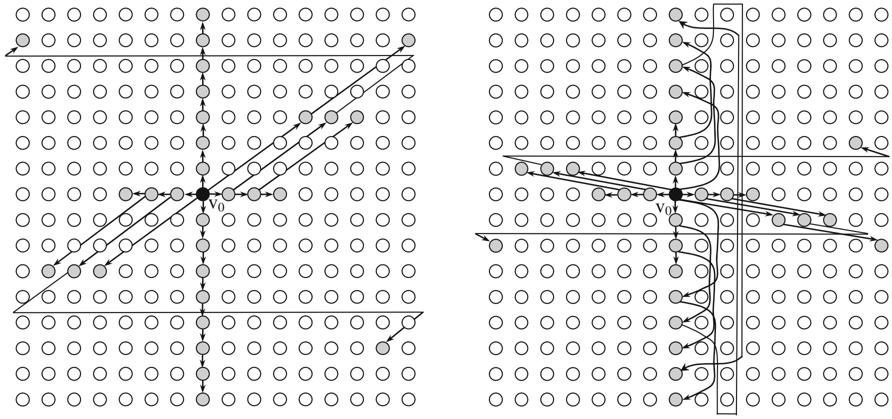

Fig. 4 Broadcast of a message from node v_0 (in black) through rows and columns in a 6-mesh with the free link $(4, 3)$ (left) an *8*-mesh with the free links $(4, -1)$ and $(0, 4)$ (right). The number of nodes is $N = 16 \times 16 = 256$ and the number of steps for all-to-all communication in 6-mesh and *8*-mesh is 11 and 6, respectively

two distinct tables are needed for the virtual rows and the virtual columns. The routing logic is the same as in *4*-meshes.

When distributing data in rows the basic normalized message length is 1, but when distributing data in columns, the basic normalized message length increases to X since the data from a whole row is being transferred instead of from a single node's. It is therefore advantageous to use the free link of a 6-mesh for transfers in virtual columns to decrease the transfer time. In the same manner, mapping multi-processor computers so that the multiple processors span columns instead of rows provides also decreases the total transfer time since the larger messages transferred in the columns travel over the faster bus link between multiple processors in a computer.

A broadcast of a message from the initial node (black) to all of the nodes of its virtual row and column is denoted by links and shown in Figure 4. The number of steps for an all-to-all algorithm in the 6-mesh and *8*-mesh, is 11 and 6, respectively. Note that the same number of steps is needed if a single column and row is added to the system $N = 17 \times 17 = 289$, which has now an odd number of nodes in columns and rows.

3.4 Comparison of performance results

A simple model of a communication channel was used in our analysis and is based on the latency and bandwidth of the channel [9]. The latency is the delay between the beginnings of the sending and receiving procedures while the bandwidth is the amount of data transferred in a time unit. The communication time for a single message is t_c = latency + message size/bandwidth, and the communication time for a global communication is $t_g = steps \times latency + data\ volume/bandwidth$, where *data volume* represents the sum of maximal message sizes over all communication steps.

Table 1 All-to-all *communication steps/data volume* with the *diameter* in brackets

N	hypercube	*4*-mesh	*8*-mesh
$16=4\times4$	4/15 (4)	4/10 (4)	2/5 (2)
$64=8\times8$	6/63 (6)	8/36 (8)	4/18 (3)
$256=16\times16$	8/255 (8)	16/136 (16)	6/102 (5)
$1024=32\times32$	10/1023 (10)	32/528 (32)	10/462 (7)

The routing procedure has an impact on the number of communication steps and data volume, both of which have a proportional impact on the communication time.

Hypercubes, *4*-meshes, and d-meshes have been compared regarding the number of communication steps and data volume needed by all-to-all algorithms. To be able to directly compare the above topologies, the meshes were of size $N = X \times X = 2^d$, where d is the dimension of the hypercube. The normalized size of a single message was equal to 1 and the communication can run concurrently in both directions.

In a hypercube, $d = \log N$, which is the number of communication steps for all-to-all communication. The data volume of a global communication in hypercubes is a sum of partial data volumes in each dimension, which is equal to $2^0+2^1+2^2+\cdots+2^{d-1} = 2^d - 1$.

From the given description of all-to-all communication in toroidal *4*-meshes it follows that the number of communication steps is equal to $X/2$ for rows and columns, respectively. In rows, only a single message is transferred in each step on each horizontal link. In columns, the merged messages from the row has the length X. The data volume for a *4*-mesh is therefore $(X + X^2)/2$.

In optimal d-meshes, f free links can be observed for the right part of the virtual row together with the links (1,0) and (0,1). Supposing that $d = 2f$ and that the virtual row can be divided into f equidistant parts, the number of required communication steps needed for gathering the messages from all nodes in a virtual row is $X/f = 2X/d$ and the same for columns, which gives the minimum number of communication steps $4X/d$. It is difficult to generally assess the total data volume and the data volumes for every step.

Comparing expressions for the number of all-to-all steps for hypercubes and d-meshes, we get: $d = \log N < 4X/d = 4\sqrt{N}/d$. Knowing that $N = 2^d$ we obtain the simplified expression $d^4 < 2^{d+4}$, which is solved for $d < 8$. Hypercubes need more communication steps than d-meshes if the number of nodes is less than 512.

Table 1 summarizes above results for the number of *communication steps/data volume* needed in all-to-all communication for hypercubes, toroidal *4*-meshes and *8*-meshes. The theoretical minimum is given in brackets by the network *diameter*. Note that for most networks with a number of nodes not equal to a power of 2, optimal d-meshes also outperform hypercubes in the number of communication steps.

4 Conclusion

We have shown that d-meshes can be applied successfully in molecular dynamics simulation. They compete with hypercubes and other interconnection topologies [21,22] in terms of the number of required steps and data volume needed in all-to-all communication. Other advantages of d-meshes are in their scalability, regularity, isomorphism, and layered structure. Since they are based on routing tables, they have potential for dynamic reconfiguration and fault tolerance. We also suggest their use in existing computing clusters in order to improve communication performance. New collective communication algorithms will be investigated in the future to exploit the remaining potential of optimal d-meshes.

Acknowledgements The authors acknowledge the financial support from the state budget by the Slovenian Research Agency under grants Nos. P2-0095 and P1-0002.

References

1. D. Janežič, Cell. Mol. Biol. Lett. **7**, 78 (2002)
2. V. Spichak, Annali Di Geofisica **44**, 273 (2001)
3. R. Hren, Phys. Med. Biol. **43**, 1449 (1998)
4. P. Trunk, B. Gersak, R. Trobec, Comput. Biol. Med, **33**, 203 (2003)
5. G. Golub, J.M. Ortega, Scientific Computing, an Introduction with Parallel Computing (Academic Press, Inc., San Diego 1993)
6. U. Borštnik, M. Hodoščček, D. Janežič, J. Chem. Inf. Comput. Sci. **44**, 359 (2004)
7. R. Trobec, M. Šterk, M. Praprotnik, D. Janežič, Int. J. Quant. Chem. **96**, 530 (2004)
8. N.R. Adiga et al., IBM J. Research Devel. **49**, 265 (2005)
9. V. Kumar, A. Grama, A. Gupta, G. Karypis, Introduction to Parallel Computing – Design and Analysis of Algorithms (The Benjamin/Cummings Publishing Company, Inc., Redmond City, 1994)
10. L.D. Aronson, Networks and Routing—A Survey (Faculteit der Technische Wiskunde en Informatica, Delft, 1994)
11. I. Jerebic, R. Trobec, Information Processing Letters **43**, 285–291 (1992)
12. Y. Yang, J. Wang, in *Proceedings of 5th IEEE International Symposium on High-Performance Computer Architecture (HPCA- 5) (IEEE, Orlando, FL, 1999)*, pp. 290–299
13. R. Trobec, Parallel Computing **26**, 1945 (2000)
14. D. Heerman, A. Burkitt, Parallel Algorithms in Computational Science (Springer-Verlag, Berlin, 1991)
15. R. Trobec, I. Jerebic, D. Janežič, Parallel Computing **19**, 1029 (1993)
16. D. Janežič, M. Praprotnik, F. Merzel, J. Chem. Phys. **122**, 174101 (2005)
17. B.R. Brooks, M. Hodoščček, Chem. Design Automat. News. **7**, 16 (1992)
18. U. Borštnik, M. Hodoščček, D. Janežič, Croat. Chem. Acta **78**, 211 (2005)
19. Diestel, Graph Theory, (Springer-Verlag, New York, 2000)
20. U. Jovanovič, R. Trobec, in: Parallel Numerics ’02, eds. R.Trobec, P. Zinterhof, M. Vajteric, A. Uhl, (University of Salzburg & Jozef Stefan Institute, Salzburg, Ljubljana, 2002) pp. 109–121
21. K. Kutnar, U. Borštnik, D. Marušič, D. Janežič, J. Math. Chem. doi:10.1007/s10910-008-9412-5
22. A.E. Vizitiu, M.V. Diudea, S. Nikolić, D. Janežič, J. Chem. Inf. Model. **46**, 2574 (2006)