Scheduling Divisible Loads in Gaussian, Mesh and Torus Network of Processors

Zhemin Zhang and Thomas G. Robertazzi

Department of Electrical and Computer Engineering
Stony Brook University, Stony Brook, NY 11794, USA

***Abstract*—In this paper, we propose a novel analysis method for divisible load scheduling in mesh, torus and *Gaussian network*, a new type of interconnection network that has the same node degree as the mesh and torus, but shorter network diameter and shorter average hop distances under equal network size. The divisible scheduling in these three networks are uniformly formulated as the *Maximum Finish Time Minimization (MFTM)* problem. It involves minimizing the makespan of the load distribution and processing. The MTFM problem, a relaxed MFTM problem, a linear programming problem version and a heuristic algorithm are described and solved. The first three of these problems have identical solutions. The heuristic algorithm is close in performance to the optimal solution, significantly outperforms the previously described dimensional algorithm, and has much wider application range than the previously proposed phase algorithm.**

***Index Terms*—Divisible load scheduling, Gaussian network, mesh, torus, maximum finish time minimization, linear programming.**

I. Introduction

A divisible load is a computational load that can be divided into any number of arbitrary small fractions, which are independent and can be processed in parallel. The divisible load model is a good approximation of tasks that require large number of identical, low-granularity computations, thus has been proposed for a wide range of scientific and engineering data processing, such as image processing, matrix multiplication, fast-fourier-transformation (FFT), video encoding/decoding, stereo matching, etc. [1]- [5].

The basic linear divisible load model assumes that the processing time of divisible load on a single processor and the transmission time of divisible load from one processor to the other are both proportional to the divisible load size [6] [7] [8]. In general, the processing time of a unit divisible load on a standard processor and the transmission time of a unit divisible load through a link with standard data rate are denoted as T_{cp} and T_{cm}, respectively. The aim of divisible load scheduling is to minimize the processing time by distributing divisible load among multiple processors which are interconnected by a specific network topology. The processing speed of these processors can be either homogeneous or heterogenous, as are the data rates of links in the network [9] [10].

Over the past two decades, there has been extensive research in the literature on scheduling divisible load in a variety of network topologies, such as daisy chain, bus, tree, hypercube, mesh and torus [6]- [8], [11]- [16]. In [6], [7], [8] and [11], the optimal solution was obtained for divisible load scheduling in daisy chain, tree, bus and hypercube, respectively. The performance limit of mesh and torus in scheduling divisible load was provided in [12]. The dimensional algorithm was proposed for N-dimensional mesh and torus in [13], which decomposes an N-dimensional mesh (or torus) into linearly connected $N-1$-dimensional meshes (or tori). Based on the dimensional algorithm, two pipeline algorithms are proposed in [14] to accelerate load distribution. In [15] and [16], the phase algorithm was proposed for torus and 3D-mesh, which divides the load distribution into several phases. In each phase, the active processors, i.e., processors that finish receiving load, and can start distributing load to the other processors, are carefully selected such that the load distribution in the next phase will not encounter link contention. The phase algorithm considers the startup time, i.e., the time for connection establishment between two processors in the network, which is also studied in [17]- [19]. In [20]- [22], the multi-installment scheme was proposed in divisible load scheduling, which allows processors to start processing and distributing load earlier. The scenario of multiple divisible load sources was studied in [23]- [25], where load distribution originates from multiple processors in the network.

In this paper, we propose a novel analysis method for divisible load scheduling, and adopt it in mesh and torus. As will be seen in Section V, the algorithm based on our proposed novel analysis method significantly outperforms the previously proposed dimensional algorithm in [13] due to its higher efficiency in utilizing links in the network to distribute load, and has much wider application range than the phase algorithm in [16], which is applicable only to torus with 5^N nodes. In addition to the mesh and torus, we also study divisible load scheduling in processors interconnected by a new type of network topology, called a *Gaussian network*, which is proposed in [26]. The Gaussian network has an equal node degree as the mesh and torus, but shorter average hop distance and network diameter than the latter two network topologies under equal network size [27]- [30]. In [28], the Gaussian network has been demonstrated to be a promising candidate for on-chip network, which outperforms on-chip mesh and torus networks in terms of communication bandwidth and latency. Moreover, by utilizing the underlying Hamiltonian cycles in the Gaussian network, a bufferless routing algorithm has been proposed for the optical Gaussian macrochip, which is a chip-scale optical network architecture adopting the Gaussian network. The optical Gaussian macrochip significantly improves the power efficiency, supports much higher communication bandwidth, and achieves much lower average packet delay compared with optical macrochips adopting other network topologies, such as mesh, torus, Clos and fully connected networks [29] [30]. Divisible load scheduling in the Gaussian network is studied along with mesh and tori in this paper due to the fact that the Gaussian network has the same node degree as the mesh

and the torus so that the divisible load scheduling in these three networks can be uniformly formulated as the same optimization problem under our proposed analysis method, as will be discussed in Section IV.

For presentational convenience, the terminologies of processor and node will be used interchangeably in the rest of this paper. We assume homogeneous processing speed and data rate for all the nodes and links, respectively, in the network, and startup time is ignored in our paper. Besides, we study the scenario that the load distribution originates from only one node in the network, and nodes can start processing and distributing load only after it finishes receiving load from its neighbors. Since the Gaussian network is recently proposed, and its interconnection is not as widely known as the mesh and torus, we begin our discussion with the Gaussian network, and formulate the divisible load scheduling in a Gaussian network as an optimization problem, denoted as *maximum finish time minimization (MFTM)* problem, in which we record the time that each node finishes processing the load, i.e., the *finish time* of each node. The object of the MFTM problem is to minimize the maximum finish time of all nodes in the network. By relaxing the constraints of the MFTM problem, we obtain the relaxed MFTM problem, which is further transformed into the *finish time minimization (FTM)* problem. We prove that these three problems have an equal optimal solution, and design an optimization algorithm based on linear programming, denote as the *LP-based* algorithm, for the FTM problem. Considering the high time complexity of the LP-based algorithm, we further propose a heuristic algorithm for the FTM problem. After the discussion on the Gaussian network, we will extend our analysis to mesh and torus. As mentioned above, the divisible load scheduling in these two networks can be formulated as an MFTM problem as well, which can still be transformed into the FTM problem, and our proposed LP-based algorithm and heuristic algorithm also apply to divisible load scheduling in mesh and torus networks.

The rest of the paper proceeds as follows. Section II introduces the Gaussian network, and some of its related properties to be used in the following parts of the paper. In Section III, we formulate divisible load scheduling in Gaussian network as the maximum finish time minimization (MFTM) problem, transform it into the finish time minimization (FTM) problem, which has equal optimal solution to the MFTM problem, and propose an optimal algorithm based on linear programming, denoted as LP-based algorithm, and a heuristic algorithm for the FTM problem. We extend our proposed MFTM problem formulation to mesh and torus in Section IV. In Section V, we compare the performance of the heuristic algorithm with the LP-based algorithm, dimensional algorithm, pipeline algorithms and phase algorithm in terms of speedup in Gaussian networks, meshes and tori with respect to different network sizes. Finally, we conclude the paper in Section VI.

II. Gaussian Networks

In this section, we briefly introduce the Gaussian network, and some of its related properties, which will be useful in the discussion of divisible load scheduling in the Gaussian network.

A. Mathematical Background

In this subsection, we provide related mathematical backgrounds, which are necessary for introducing the Gaussian network.

A Gaussian network is a network topology defined by Gaussian integers. Gaussian integers are a subset of complex numbers with integral real and integral imaginary parts, which is defined as

$$\mathbf{Z}[\mathbf{i}] = \{\omega = x + y\mathbf{i} \mid x, y \in \mathbf{Z}\}$$

where $\mathbf{Z}$ is the set of integers, and $\mathbf{i}^2 = -1$.

Given a non-zero Gaussian integer $a + b\mathbf{i}$, and two Gaussian integers ω and ω', if there exist a Gaussian integer $a' + b'\mathbf{i}$ such that

$$\omega - \omega' = (a' + b'\mathbf{i})(a + b\mathbf{i})$$

we say that ω and ω' are *congruent modulo* $a+b\mathbf{i}$, which is denoted as

$$\omega \equiv \omega' \bmod a + b\mathbf{i}$$

and that ω and ω' belong to the same *congruence class modulo* $a+b\mathbf{i}$. For instance, $(6+\mathbf{i})-(-1+2\mathbf{i}) = (1-\mathbf{i})(4+3\mathbf{i})$, therefore, $6+\mathbf{i}$ and $-1+2\mathbf{i}$ belong to the same congruence class modulo $4+3\mathbf{i}$. Congruence modulo is an equivalence relation, which has symmetry, reflectivity and transitivity. It has been shown that for Gaussian integer $a+b\mathbf{i}$, there are a^2+b^2 different congruence classes modulo $a+b\mathbf{i}$ in total, and any given Gaussian integer belongs to one of these a^2+b^2 congruence classes [27] [31]. Next, we define Gaussian network by the introduced terminologies above.

B. Network Interconnection

In this subsection, we discuss Gaussian network interconnections and some of its properties.

A Gaussian network defined by a non-zero Gaussian integer $a + b\mathbf{i}$, denoted as $G_{a+b\mathbf{i}}$, has $a^2 + b^2$ nodes, each represented by a Gaussian integer that belongs to a distinguished congruent class modulo $a + b\mathbf{i}$, and the items of Gaussian integer and node will be used interchangeably in the rest of our paper.

Given two nodes ω_1 and ω_2 in $G_{a+b\mathbf{i}}$, there exists an edge between ω_1 and ω_2 if and only if

$$\omega_1 - \omega_2 \equiv \mathbf{i}^j \bmod a + b\mathbf{i} \tag{1}$$

where $j = 0, 1, 2$ and 3, and we say that ω_1 is *neighbor* j of ω_2.

According to Eq. (1), all nodes in a Gaussian network are symmetric [26], and a node in Gaussian networks has as many as 4 neighbors. In addition, if a Gaussian network has more than 4 nodes, any node in it has 4 neighbors, i.e., the node degree is 4 [26]. Beside, Eq. (1) indicates

$$\omega_2 - \omega_1 \equiv -\mathbf{i}^j \bmod a + b\mathbf{i}$$

therefore, ω_2 is neighbor $\mathrm{mod}_4(j+2)$ of ω_1, where $\mathrm{mod}_4(j+2)$ is $j+2$ modulo 4. Note that Gaussian networks with higher node degree than 4 can be generalized [32], but we only consider those with node degree of 4 as they have equal node degree as the mesh and torus, and the divisible load scheduling in these three networks can be uniformly formulated as the *Maximum Finish Time Minimization (MFTM)* problem, to be seen in the following of the paper. Fig. 1 is an example of Gaussian network $G_{4+3\mathbf{i}}$ with 25 nodes, which are placed in two adjacent meshes in the complex plane, and node $6+\mathbf{i}$ is neighbor 2 of node $2\mathbf{i}$ in $G_{4+3\mathbf{i}}$ since $(6+\mathbf{i}) - 2\mathbf{i} \equiv \mathbf{i}^2 \bmod 4+3\mathbf{i}$, indicating node $2\mathbf{i}$ is neighbor 0 of node $6+\mathbf{i}$. It has been proved in [26] that Gaussian networks $G_{\pm a \pm b\mathbf{i}}$ and $G_{\pm b \pm a\mathbf{i}}$ are isomorphic, therefore, without loss of generality, we assume that $a \geq b \geq 0$ in the rest of our paper.

The distance between two nodes, say, ω_1 and ω_2, in $G_{a+b\mathbf{i}}$, denoted as $\mathbf{D}(\omega_1, \omega_2)$, is given as follows.

$$\mathbf{D}(\omega_1, \omega_2) = \min\{|x| + |y|, x + y\mathbf{i} \equiv (\omega_1 - \omega_2) \bmod a + b\mathbf{i}\} \tag{2}$$

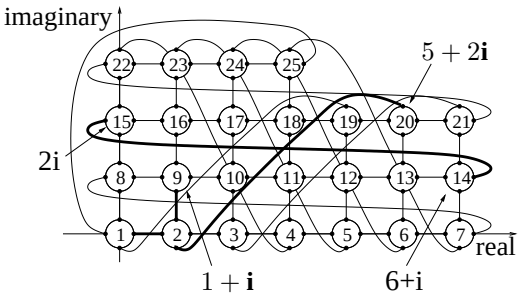

Fig. 1. Gaussian network $G_{4+3\mathbf{i}}$ with nodes placed in two adjacent meshes.

That is, $\mathbf{D}(\omega_1, \omega_2)$ equals the minimum $|x| + |y|$, such that $x + y\mathbf{i}$ and $\omega_1 - \omega_2$ belong to the same congruence class modulo $a + b\mathbf{i}$.

For example, in Fig. 1, $\mathbf{D}(0, 1 + \mathbf{i}) = |1| + |1| = 2$, and $\mathbf{D}(1 + \mathbf{i}, 5 + 2\mathbf{i})$ is 2 as well since that $2\mathbf{i} \equiv (1 + \mathbf{i}) - (5 + 2\mathbf{i}) \bmod 4 + 3\mathbf{i}$, and when $\omega = 2\mathbf{i}$, $|x| + |y|$ is minimized in Eq. (2).

The network diameter of Gaussian network $G_{a+b\mathbf{i}}$ is a when $a+b$ is even, and is $a-1$ when $a+b$ is odd, and its average hop distance is given in Lemma 1 [26]. The Gaussian network has the same node degree as the mesh and torus, and is advantageous over the latter two network topologies in terms of average hop distance and network diameter. For example, Gaussian network $G_{4+3\mathbf{i}}$ has 25 nodes, and its network diameter is 3, while the network diameters of a mesh and a torus of the same network size are 8 and 4, respectively. In Table 1 and 2, we list the average hop distances and network diameters of Gaussian network, mesh and torus with respect to different network sizes, which show that Gaussian network always has shorter average hop distance and network diameter than mesh and torus under equal network size.

Lemma 1: The average hop distance of Gaussian network $G_{a+b\mathbf{i}}$ is

$$\begin{cases} \frac{3a(a^2+b^2)+2b(b^2-1)}{6(a^2+b^2-1)} & \text{if } a+b \text{ is even} \\ \frac{3a(a^2+b^2-1)+2b(b^2-1)}{6(a^2+b^2-1)} & \text{if } a+b \text{ is odd} \end{cases}$$

TABLE 1
AVERAGE HOP DISTANCE COMPARISON AMONG GAUSSIAN NETWORK, MESH AND 2D-TORUS

	Network size		
	25 nodes	100 nodes	400 nodes
Gaussian network	2.3333	4.7475	9.4536
Mesh	3.2000	6.6667	13.3333
torus	2.4000	5	10

TABLE 2
NETWORK DIAMETER COMPARISON AMONG GAUSSIAN NETWORK, MESH AND 2D-TORUS

	Network size		
	25 nodes	100 nodes	400 nodes
Gaussian network	3	8	16
Mesh	8	18	38
torus	4	10	20

A Gaussian network is optimal if it accommodates the most number of nodes among all Gaussian networks with the same network diameter. It has been proved in [26] that Gaussian network $G_{a+b\mathbf{i}}$ is optimal if and only if $a = b+1$, and its network diameter is b.

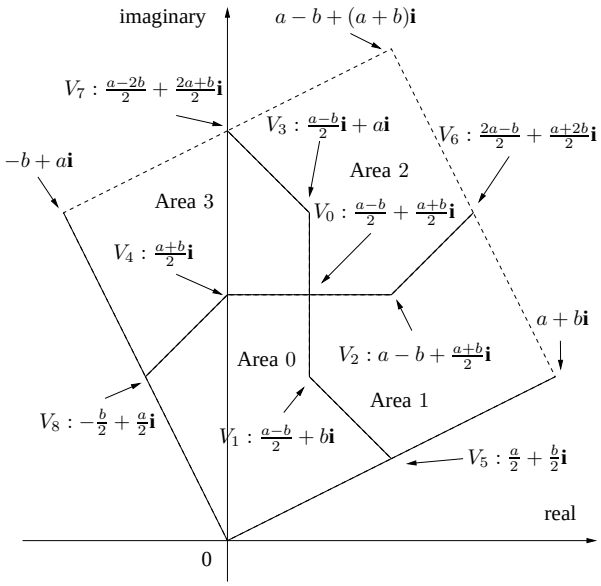

Fig. 2. Half-open square $S_{a+b\mathbf{i}}$, which excludes dash-lined boundary, is decomposed into four areas. The vertex coordinates of each area are labeled.

C. Symmetric Node Placement for Gaussian Network

In [28], it has been pointed out that any group of a^2+b^2 Gaussian integers in the complex plane that consists of a complete collection of a^2+b^2 congruent classes modulo $a+b\mathbf{i}$ can be used to represent nodes in Gaussian network $G_{a+b\mathbf{i}}$, and the constructed networks are isomorphic as long as they are interconnected by Eq. (1) [28]. Besides, given a nonzero Gaussian integer $a + b\mathbf{i}$, there are $a^2 + b^2$ Gaussian integers in a *half-open square* $S_{a+b\mathbf{i}}$ defined by

$$S_{a+b\mathbf{i}} = \{(u + v\mathbf{i})(a + b\mathbf{i}) | 0 \le u, v < 1\}$$

as shown in Fig. 2, which excludes dash-lined boundary. These Gaussian integers each belong to a distinguished congruent class modulo $a + b\mathbf{i}$, therefore, can be used to represent all the nodes in Gaussian network $G_{a+b\mathbf{i}}$ [26], and Fig. 3 is an example of node placement in half-open square $S_{4+3\mathbf{i}}$ for Gaussian network $G_{4+3\mathbf{i}}$, which is isomorphic to the node placement in two adjacent meshes in Fig. 1.

To explore the symmetry of Gaussian network $G_{a+b\mathbf{i}}$, a node placement in a *half-open polygon* $P_{a+b\mathbf{i}}$ is proposed in [28]. The half-open polygon $P_{a+b\mathbf{i}}$ is constructed by firstly decomposing the half-open square $S_{a+b\mathbf{i}}$ into four non-overlapping areas, as shown in Fig. 2, where the vertex coordinates of the four areas are labelled. The edges V_0V_{j+1} and $V_{j+1}V_{j+5}$ belong to Area j, where $j = 0, 1, 2$ and 3, and V_0 belongs to Area 1. These four areas are then shifted according to Eq. (3), where ω_s and ω_p are the coordinates of the point before and after the shifting. That is, every point ω_s in $S_{a+b\mathbf{i}}$ is mapped to ω_p in $P_{a+b\mathbf{i}}$ by Eq. (3). As each Gaussian integer in $S_{a+b\mathbf{i}}$ and its mapped Gaussian integer in $P_{a+b\mathbf{i}}$ belong to the same congruence class modulo $a + b\mathbf{i}$, all Gaussian integers in $P_{a+b\mathbf{i}}$ also consists of a complete collection of $a^2 + b^2$ congruence classes modulo $a+b\mathbf{i}$, and can be used to represent nodes in $G_{a+b\mathbf{i}}$.

$$\omega_p = \begin{cases} \omega_s & \text{if } \omega_s \text{ is in Area 0} \\ \omega_s - (a + b\mathbf{i}) & \text{if } \omega_s \text{ is in Area 1} \\ \omega_s - (1 + \mathbf{i})(a + b\mathbf{i}) & \text{if } \omega_s \text{ is in Area 2} \\ \omega_s - \mathbf{i}(a + b\mathbf{i}) & \text{if } \omega_s \text{ is in Area 3} \end{cases} \quad (3)$$

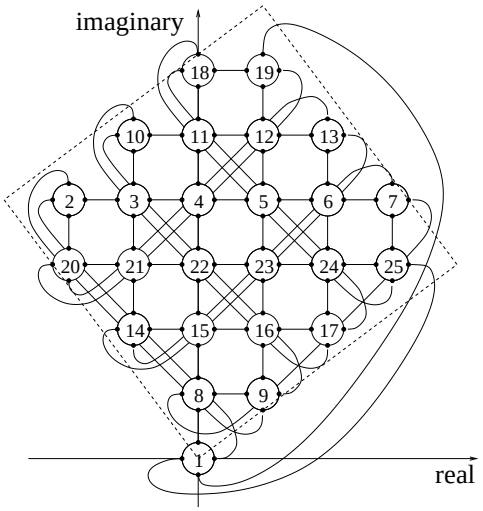

Fig. 3. Nodes in Gaussian network $\mathbf{G}_{4+3\mathbf{i}}$ placed on half-open square $S_{4+3\mathbf{i}}$.

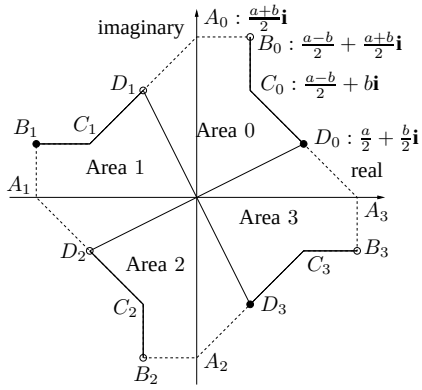

Fig. 4. Half-open polygon for node placement of $G_{a+b\mathbf{i}}$, which excludes dash-lined boundary and points depicted by hollow dots on the boundary, and is denoted as $P_{a+b\mathbf{i}}$. Without the solid points on the boundary, i.e., B_1, D_0 and D_3, $P_{a+b\mathbf{i}}$ is 4-fold rotational symmetric, i.e., it can overlap with itself after being rotated by 90 degrees. The coordinates of A_0, B_0, C_0 and D_0 are $\frac{a+b}{2}\mathbf{i}$, $\frac{a-b}{2} + \frac{a+b}{2}\mathbf{i}$, $\frac{a-b}{2} + b\mathbf{i}$ and $\frac{a}{2} + \frac{b}{2}\mathbf{i}$, respectively. The remaining vertex coordinates can be obtained by the rotational symmetry of $P_{a+b\mathbf{i}}$.

As shown in Fig. 4, the obtained half-open polygon $P_{a+b\mathbf{i}}$ after shifting is a polygon including partial points on its boundary, and we use solid line and dash line to depict the boundary that $P_{a+b\mathbf{i}}$ includes and excludes, respectively. In addition, the common point of the solid-lined and dash-lined boundary is depicted by a solid dot if it belongs to $P_{a+b\mathbf{i}}$, otherwise, it is depicted by a hollow dot, and we can see that B_1, D_0 and D_3 belong to $P_{a+b\mathbf{i}}$ in Fig. 4, which are named as *solid points*. An important property of the half-open polygon is that if the solid points B_1, D_0 and D_3 are removed, the half-open polygon is 4-fold rotational symmetric, i.e., it can overlap itself after being rotated by 90 degrees centering at the origin of the complex plane. Therefore, we label the coordinates of only A_0, B_0, C_0 and D_0 in Fig. 4, which are $\frac{a+b}{2}\mathbf{i}$, $\frac{a-b}{2} + \frac{a+b}{2}\mathbf{i}$, $\frac{a-b}{2} + b\mathbf{i}$ and $\frac{a}{2} + \frac{b}{2}\mathbf{i}$, respectively. The remaining vertex coordinates can be obtained by the rotational symmetry of the half-open polygon.

It is worth mentioning that we can have different assignments for common edges of the four areas in Fig. 2, which yield different node placements for $G_{a+b\mathbf{i}}$. For example, if edges V_0V_{j+1} and $V_{j+1}V_{j+5}$ still belong to Area j, where $j = 0, 1, 2$ and 3, but V_0 belongs to Area 2, and we let V_8 belong to Area 0, shifting the areas according to Eq. 3 yields another half-open polygon, denoted as $P'_{a+b\mathbf{i}}$, in Fig. 5, where the Gaussian integers can also be used to represent nodes in $G_{a+b\mathbf{i}}$. In $P_{a+b\mathbf{i}}$ and $P'_{a+b\mathbf{i}}$, Gaussian integers belonging to the same congruence class modulo $a + b\mathbf{i}$ represent the same node in $G_{a+b\mathbf{i}}$, which maintains the neighboring relationship among nodes, i.e., neighbor j of a given node in $G_{a+b\mathbf{i}}$ will still be represented by Gaussian integers belonging to the same congruence class modulo $a + b\mathbf{i}$ in $P_{a+b\mathbf{i}}$ and $P'_{a+b\mathbf{i}}$ [28]. For example, when $a+b$ is even, neighbor 2 of node $\frac{a-b}{2} + (\frac{a+b}{2} - 1)\mathbf{i}$ is $\frac{a-b}{2} + \frac{a+b}{2}\mathbf{i}$ and $-\frac{a+b}{2} + \frac{a-b}{2}\mathbf{i}$ in $P_{a+b\mathbf{i}}$ and $P'_{a+b\mathbf{i}}$, respectively, and $\frac{a-b}{2} + \frac{a+b}{2}\mathbf{i} \equiv -\frac{a+b}{2} + \frac{a-b}{2}\mathbf{i}$ modulo $a + b\mathbf{i}$. As will be seen shortly, placing nodes in different half-open polygons reveals several properties of divisible load scheduling in Gaussian networks.

In our paper, we also place nodes of $G_{a+b\mathbf{i}}$ in $P_{a+b\mathbf{i}}$. Fig. 6 is an example of node placement for $G_{4+3\mathbf{i}}$ in $P_{4+3\mathbf{i}}$, of which the boundary is plotted in dash line. As for the node placement for an optimal Gaussian network $G_{b+1+b\mathbf{i}}$, we have the following corollary.

Corollary 1: The nodes of an optimal Gaussian network $G_{b+1+b\mathbf{i}}$ can be placed in the interior of a square, i.e., excluding its boundary, with $\pm\frac{2b+1}{2}$ and $\pm\frac{2b+1}{2}\mathbf{i}$ as the vertices in the complex plane.

Proof: In $P_{a+b\mathbf{i}}$, if a Gaussian integer, say, ω, is in the triangle $A_0B_0C_0$, $A_1B_1C_1$, $A_2B_2C_2$ or $A_3B_3C_3$, it must satisfy either $|x| \le \frac{a-b}{2}$ or $|y| \le \frac{a-b}{2}$. Therefore, when $a = b + 1$, no Gaussian integers resides in these triangles, and all Gaussian integers in $P_{a+b\mathbf{i}}$ must reside in the square with A_1, A_2, A_3 and A_4 as the vertices.

In addition, if ω is on the boundary of the square, we have $|x| + |y| = \frac{2b+1}{2}$, which is impossible since $|x| + |y|$ must be integer. Hence, there are no Gaussian integers on the boundary of the square, and all Gaussian integers must be in the interior of the square, which can be used to represent all nodes in $G_{b+1+b\mathbf{i}}$. ■

Since $G_{4+3\mathbf{i}}$ is an optimal Gaussian network, we can see from Fig. 6 that all of its nodes reside in the interior of the square with $\pm\frac{7}{2}$ and $\pm\frac{7}{2}\mathbf{i}$ as the vertices.

Next, we will formulate the divisible load scheduling problem in a Gaussian network as an optimization problem based on the introduced node placement.

In the next section, we will discuss divisible load scheduling in Gaussian networks.

III. Scheduling A Divisible Load in Gaussian Networks

In this section, we formulate the divisible load scheduling problem in Gaussian network as an optimization problem, which is denoted as *maximum finish time minimization (MFTM)* problem, and propose an optimal algorithm for it.

A. Problem Formulation

As mentioned in Section I, we assume homogeneous processing speed and link data rate in our paper. In the Gaussian network, the processing time of a unit divisible load on a single processor is denoted as T_{cp}, and we denote the transmission time of a unit divisible load through a link in the network as T_{cm}. Since every node is symmetric in Gaussian networks, without loss of generality, we assume that the load originates from the node at the origin of the complex plane, and spreads to the surrounding nodes hop by hop.

In our model, a node, say, ω, in $G_{a+b\mathbf{i}}$ can only receive load from (or send load to) its neighbors that are closer to (or further away from) the origin than itself, and its neighbor j is denoted as $n_j(\omega)$. Node ω is allowed to receive load from (or send load to)

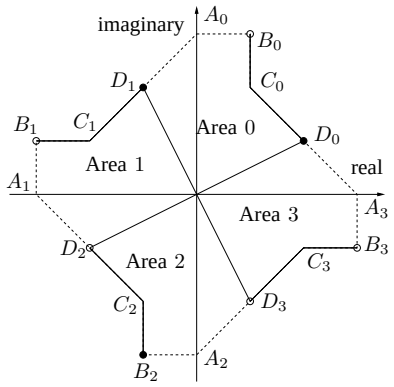

Fig. 5. Another half-open polygon, denoted as $P'_{a+b\mathbf{i}}$, for node placement of $G_{a+b\mathbf{i}}$, where B_2, D_0 and D_3 on the boundary are solid points.

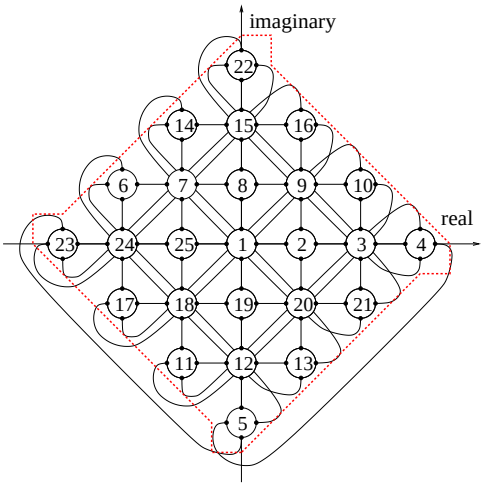

Fig. 6. Node placement of $G_{4+3\mathbf{i}}$ in $P_{4+3\mathbf{i}}$, the boundary of which is plotted in dash line.

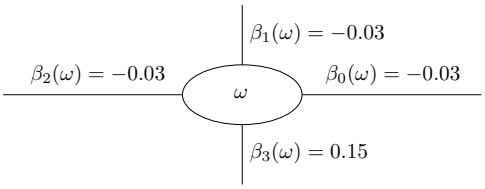

Fig. 7. Node ω receives 0.15 load from its neighbor 3, and sends 0.03 load to each of its rest 3 neighbors.

one of its neighbors for at most once, and the amount of load ω receives from neighbor $n_j(\omega)$ is denoted as $\beta_j(\omega)$. After ω finishes receiving load from all its neighbors, it starts processing and sending out load simultaneously. We denote the amount of load ω processes by itself as $\alpha(\omega)$, and the time it starts and finishes processing load as $T_s(\omega)$ and $T_f(\omega)$, respectively. For presentational convenience, if ω sends load to neighbor $n_j(\omega)$, we let $\beta_j(\omega) < 0$, which means that neighbor $n_j(\omega)$ receives $-\beta_j(\omega)$ load from ω. As mentioned in Section II-B, ω is also neighbor $\text{mod}_4(j+2)$ of $n_j(\omega)$, and we have that $\beta_j(\omega) = -\beta_k(n_j(\omega))$, where $k = \text{mod}_4(j+2)$. Take Fig. 7 for example, node ω receives 0.15 load from its neighbor 3, and sends 0.03 load to each of the rest neighbors.

By the definition of $\beta_j(\omega)$, we have that $\beta_j(\omega) \geq 0$, $\beta_j(\omega) \leq 0$ and $\beta_j(\omega) = 0$ when $\mathbf{D}(0,\omega) > \mathbf{D}(0,n_j(\omega))$, $\mathbf{D}(0,\omega) < \mathbf{D}(0,n_j(\omega))$ and $\mathbf{D}(0,\omega) = \mathbf{D}(0,n_j(\omega))$, respectively, based on which we divide all $\beta_j(\omega)$s into 3 sets, U_β^+, U_β^- and U_β^0, as follows to facilitate the problem formulation of divisible load scheduling in Gaussian network.

$$U_\beta^+ = \{\beta_j(\omega) | \mathbf{D}(0,\omega) > \mathbf{D}(0,n_j(\omega))\}$$
$$U_\beta^- = \{\beta_j(\omega) | \mathbf{D}(0,\omega) < \mathbf{D}(0,n_j(\omega))\}$$
$$U_\beta^0 = \{\beta_j(\omega) | \mathbf{D}(0,\omega) = \mathbf{D}(0,n_j(\omega))\}$$

Clearly, $\beta_j(\omega) \geq 0$ if $\beta_j(\omega) \in U_\beta^+$, $\beta_j(\omega) \leq 0$ if $\beta_j(\omega) \in U_\beta^-$ and $\beta_j(\omega) = 0$ if $\beta_j(\omega) \in U_\beta^0$. In addition, since $\beta_j(\omega) = -\beta_k(n_j(\omega))$, where $k = \text{mod}_4(j+2)$, $\beta_j(\omega) \in U_\beta^+$ if and only if $\beta_k(n_j(\omega)) \in U_\beta^-$, vice versa.

The divisible load scheduling problem in Gaussian networks is then formulated as the optimization problem in Table 3, which is denoted as *maximum finish time minimization (MFTM)* problem.

The objective of the MFTM problem is to minimize the maximum finish time of all nodes in the network. Constraint (4) means that it takes $\alpha(\omega)T_{cp}$ time for node ω to process its load. Constraint (5) and (6) indicates that the load originates from node 0, and a node starts processing the load when it finishes receiving all load from its neighbors. Note that if a node receives no load from its neighbors, its starting time is set as 0, as stated by constraint (7). The next 4 constraints originate from our rules of $\beta_j(\omega)$, as discussed above. Constraint (12) and (13) state that the load node ω keeps for itself equals the difference between the load it receives and sends out, and that the total load is 1, implying that $\sum \alpha(\omega) = 1$. The last constraint means that a node can not send out more load than it receives.

TABLE 3
MAXIMUM FINISH TIME MINIMIZATION (MFTM) PROBLEM FORMULATION FOR DIVISIBLE LOAD SCHEDULING IN A GAUSSIAN NETWORK

Minimize: $\max\{T_f(\omega), \omega \in G_{a+b\mathbf{i}}\}$

Subject to:

$$T_f(\omega) = T_s(\omega) + \alpha(\omega)T_{cp} \quad (4)$$
$$T_s(0) = 0 \quad (5)$$
$$T_s(\omega) = \max\{T_s(n_j(\omega)) + \beta_j(\omega)T_{cm} | \beta_j(\omega) > 0\} \quad (6)$$
$$T_s(\omega) = 0, \text{ if } \forall j \in \{0,1,2,3\}, \ \beta_j(\omega) = 0 \quad (7)$$
$$\beta_j(\omega) \geq 0, \text{ if } \beta_j(\omega) \in U_\beta^+ \quad (8)$$
$$\beta_j(\omega) \leq 0, \text{ if } \beta_j(\omega) \in U_\beta^- \quad (9)$$
$$\beta_j(\omega) = 0, \text{ if } \beta_j(\omega) \in U_\beta^0 \quad (10)$$
$$\beta_j(\omega) = -\beta_k(n_j(\omega)), \text{ if } k = \text{mod}_4(j+2) \quad (11)$$
$$\alpha(\omega) = \sum_{j=0}^{3} \beta_j(\omega), \text{ if } \omega \neq 0 \quad (12)$$
$$\alpha(0) = 1 + \sum_{j=0}^{3} \beta_j(0) \quad (13)$$
$$\alpha(\omega) \geq 0 \quad (14)$$

We say that

$$U = U_\beta^+ \cup U_\beta^- \cup U_\beta^0 \cup \{\alpha(\omega), T_s(\omega), T_f(\omega) | \omega \in G_{a+b\mathbf{i}}\}$$

is a feasible solution of MFTM problem if elements in U satisfy all the constraints of the MFTM problem, and the optimal solution is denoted as U^*. It is worthwhile to mention that given the location of source node, we can identify U_β^+, U_β^- and U_β^0 for mesh and torus, with which divisible load scheduling in these two networks can also be formulated as the MFTM problem in Table 3, as will be seen in Section IV.

Before solving the MFTM problem, we firstly analyze the characteristics of U^* by exploring the symmetry of Gaussian network. We have the following lemma and corollaries.

Lemma 2: In U^*, if ω_1 and $\omega_2 = \omega_1 \cdot \mathbf{i}$ are both in $P_{a+b\mathbf{i}}$, $\beta_j(\omega_1) = \beta_k(\omega_2)$, where $k = \mathrm{mod}_4(j+1)$.

Proof: As mentioned above, nodes in $G_{a+b\mathbf{i}}$ can also be placed in $P'_{a+b\mathbf{i}}$, where D_0, D_1 and B_2 are solid points, as well as in $P_{a+b\mathbf{i}}$. For convenience, if we place nodes in $P'_{a+b\mathbf{i}}$, the load ω sends to $n_j(\omega)$ is denoted as $\beta'_j(\omega)$, and the optimal solution for the corresponding MFTM problem is denoted as U'^*.

We notice that after being counterclockwisely rotated by 90 degrees, $P_{a+b\mathbf{i}}$ will overlap with $P'_{a+b\mathbf{i}}$, which means that if ω_1 is in $P_{a+b\mathbf{i}}$, $\omega_2 = \omega_1 \cdot \mathbf{i}$ is in $P'_{a+b\mathbf{i}}$ and $\beta_j(\omega_1) = \beta'_k(\omega_2)$, where $k = \mathrm{mod}_4(j+1)$. Besides, as Gaussian integers belonging to the same congruence class modulo $a+b\mathbf{i}$ in $P_{a+b\mathbf{i}}$ and $P'_{a+b\mathbf{i}}$ represent the same node in $G_{a+b\mathbf{i}}$, which maintains the neighboring relationship among nodes, the load that a node in $G_{a+b\mathbf{i}}$ sends to its neighbor k is independent of the node placement. Therefore, if ω_2 is also in $P_{a+b\mathbf{i}}$, we have that $\beta_k(\omega_2) = \beta'_k(\omega_2)$ as $\omega_2 \equiv \omega_2$ modulo $a+b\mathbf{i}$. ■

Corollary 2: In U^*, $\beta_j(-\frac{a+b}{2} + \frac{a-b}{2}\mathbf{i}) = \beta_k(-\frac{a+b}{2} + \frac{a-b}{2}\mathbf{i})$, where $k = \mathrm{mod}_4(j+1)$, when $a+b$ is even.

Proof: By the proof of Lemma 2, since $-\frac{a-b}{2} - \frac{a+b}{2}\mathbf{i} = (-\frac{a+b}{2} + \frac{a-b}{2}\mathbf{i}) \cdot \mathbf{i}$, $\beta_j(-\frac{a+b}{2} + \frac{a-b}{2}\mathbf{i}) = \beta'_k(-\frac{a-b}{2} - \frac{a+b}{2}\mathbf{i})$, where $k = \mathrm{mod}_4(j+1)$. In addition, $-\frac{a+b}{2} + \frac{a-b}{2}\mathbf{i}$ and $-\frac{a-b}{2} - \frac{a+b}{2}\mathbf{i}$ belong to the same congruence class modulo $a+b\mathbf{i}$, therefore, $\beta_k(-\frac{a+b}{2} + \frac{a-b}{2}\mathbf{i}) = \beta'_k(-\frac{a-b}{2} - \frac{a+b}{2}\mathbf{i})$. ■

Corollary 3: In U^*, $\beta_j(\frac{a}{2} + \frac{b}{2}\mathbf{i}) = \beta_k(\frac{a}{2} + \frac{b}{2}\mathbf{i})$ and $\beta_j(\frac{b}{2} - \frac{a}{2}\mathbf{i}) = \beta_k(\frac{b}{2} - \frac{a}{2}\mathbf{i})$, where $k = \mathrm{mod}_4(j+2)$, when a and b are both even.

Proof: Since $\frac{a}{2} + \frac{b}{2}\mathbf{i} = (\frac{b}{2} - \frac{a}{2}\mathbf{i}) \cdot \mathbf{i}$ and $-\frac{b}{2} + \frac{a}{2}\mathbf{i} = (\frac{a}{2} + \frac{b}{2}\mathbf{i}) \cdot \mathbf{i}$, we have that $\beta_j(\frac{b}{2} - \frac{a}{2}\mathbf{i}) = \beta'_{k'}(\frac{a}{2} + \frac{b}{2}\mathbf{i})$ and $\beta_{k'}(\frac{a}{2} + \frac{b}{2}\mathbf{i}) = \beta'_k(-\frac{b}{2} + \frac{a}{2}\mathbf{i})$, where $k' = \mathrm{mod}_4(j+1)$ and $k = \mathrm{mod}_4(k'+1)$. In addition, $\frac{b}{2} - \frac{a}{2}\mathbf{i} \equiv -\frac{b}{2} + \frac{a}{2}\mathbf{i}$ modulo $a+b\mathbf{i}$, and $\frac{a}{2} + \frac{b}{2}\mathbf{i}$ in $P_{a+b\mathbf{i}}$ and $P'_{a+b\mathbf{i}}$ represent the same node of $G_{a+b\mathbf{i}}$, therefore, $\beta_k(\frac{b}{2} - \frac{a}{2}\mathbf{i}) = \beta'_k(-\frac{b}{2} + \frac{a}{2}\mathbf{i})$ and $\beta_{k'}(\frac{a}{2} + \frac{b}{2}\mathbf{i}) = \beta'_{k'}(\frac{a}{2} + \frac{b}{2}\mathbf{i})$. Hence, we have $\beta_j(\frac{b}{2} - \frac{a}{2}\mathbf{i}) = \beta_k(\frac{b}{2} - \frac{a}{2}\mathbf{i})$, where $k = \mathrm{mod}_4(j+2)$, and $\beta_j(\frac{b}{2} - \frac{a}{2}\mathbf{i}) = \beta_{k'}(\frac{a}{2} + \frac{b}{2}\mathbf{i})$, indicating that $\beta_j(\frac{a}{2} + \frac{b}{2}\mathbf{i}) = \beta_k(\frac{a}{2} + \frac{b}{2}\mathbf{i})$ also holds. ■

Corollary 4: In U^*, $\beta_0(x+y\mathbf{i}) = \beta_0(x-y\mathbf{i}) = \beta_1(-y+x\mathbf{i}) = \beta_1(y+x\mathbf{i}) = \beta_2(-x-y\mathbf{i}) = \beta_2(-x+y\mathbf{i}) = \beta_3(y-x\mathbf{i}) = \beta_3(-y-x\mathbf{i})$ if the Gaussian network is optimal.

Proof: Corollary 1 says that nodes in an optimal Gaussian network, say, $G_{b+1+b\mathbf{i}}$, can be placed in the interior of a square with $\pm\frac{2b+1}{2}$ and $\pm\frac{2b+1}{2}\mathbf{i}$ as the vertices. A square has 4 axes of symmetry, the symmetric points of $x+y\mathbf{i}$ with respect to these 4 axes of symmetry are $-x+y\mathbf{i}$, $x-y\mathbf{i}$, $y+x\mathbf{i}$ and $-y-x\mathbf{i}$, as shown in Fig. 8. Therefore, we have $\beta_0(x+y\mathbf{i}) = \beta_2(-x+y\mathbf{i}) = \beta_0(x-y\mathbf{i}) = \beta_1(y+x\mathbf{i}) = \beta_3(-y-x\mathbf{i})$.

Moreover, since the square is also 4-fold rotational symmetric, by Lemma 2, we have that $\beta_0(x+y\mathbf{i}) = \beta_1(-y+x\mathbf{i}) = \beta_2(-x-y\mathbf{i}) = \beta_3(y-x\mathbf{i})$. ■

With the above lemma and corollaries, we can reduce the number of independent variables in the MFTM problem. As will be seen in Section III-D, taking these dependencies among variables into consideration can improve the efficiency of our proposed optimal algorithm and heuristic algorithm. Next, we solve the MFTM problems by transforming it into other problems with an identical optimal solution.

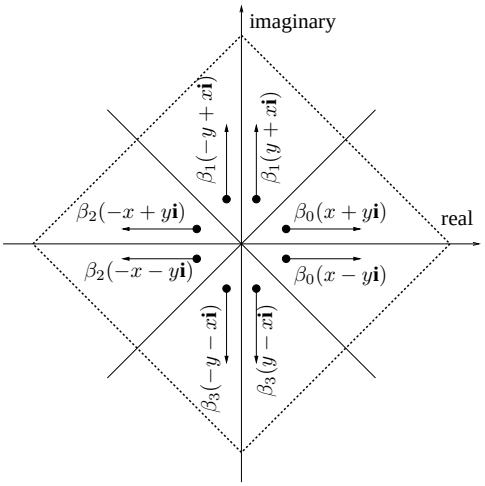

Fig. 8. The symmetry of the square renders that $\beta_0(x+y\mathbf{i}) = \beta_0(x-y\mathbf{i}) = \beta_1(-y+x\mathbf{i}) = \beta_1(y+x\mathbf{i}) = \beta_2(-x-y\mathbf{i}) = \beta_2(-x+y\mathbf{i}) = \beta_3(y-x\mathbf{i}) = \beta_3(-y-x\mathbf{i})$ in U^* when the Gaussian network is optimal.

B. Linear Relaxation of MFTM

In order to solve the MFTM problem, we first relax constraint (6) of the MFTM problem into a linear constraint as follows, and denote the new optimization problem as the *relaxed MFTM* problem.

$$T_s(\omega) \geq T_s(n_j(\omega)) + \beta_j(\omega) T_{cm}, \text{if } \beta_j(\omega) > 0 \qquad (15)$$

Since constraint (15) is relaxed from constraint (6), U^*, i.e., the optimal solution of the MFTM problem must be a feasible solution of the relaxed MFTM problem, and we have the following theorems on the optimal solution of the relaxed MFTM problem.

Theorem 1: When the relaxed MFTM problem is optimized, if $T_s(\omega) > 0$, $T_f(\omega) = T_f(0)$.

Proof: We prove the theorem by contradiction, and assume that there exist one node with different finish time from node 0, and its start time is positive. By constraint (6) and (7), we know that if $T_s(\omega) > 0$, ω must receive nonzero load from at least one of its neighbors, and that $T_f(\omega) = 0$ if $\omega \neq 0$ and $T_s(\omega) = 0$, therefore, $\max\{T_f(\omega) | \omega \in G_{a+b\mathbf{i}}\}$ is either $T_f(0)$ or $\max\{T_f(\omega) | T_s(\omega) > 0\}$.

Next, we will reduce $\max\{T_f(\omega) | \omega \in G_{a+b\mathbf{i}}\}$ by redistributing load among nodes in the network such that nodes with maximum finish time can have less load to process, and the nodes with earlier finish time process more load. For presentational convenience, we denote the start and finish time of node ω after the load redistribution as $T'_s(\omega)$ and $T'_f(\omega)$, respectively, and the load ω kept for itself after the load redistribution is denoted as $\alpha'(\omega)$.

Two cases are considered in the proof.

Case 1: $T_f(0) < \max\{T_f(\omega) | \omega \in G_{a+b\mathbf{i}}\}$.

In this case,

$$\max\{T_f(\omega) | \omega \in G_{a+b\mathbf{i}}\} = \max\{T_f(\omega), T_s(\omega) > 0\}$$

and we suppose that a node with maximum finish time is h hops away from node 0, which is denoted as ω_h, i.e.,

$$T_f(\omega_h) = \max\{T_f(\omega) | \omega \in G_{a+b\mathbf{i}}\}$$

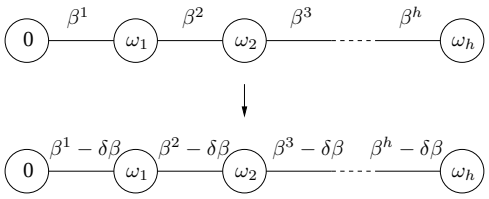

Fig. 9. Load redistribution when $T_f(0) < T_f(\omega_h) = \max\{T_f(\omega)|\omega \in G_{a+b\mathbf{i}}\}$, where $\beta^k > 0$, and β^k is reduced by $\delta\beta$ ($0 < \delta\beta < \beta^k$) for all $1 \le k \le h$.

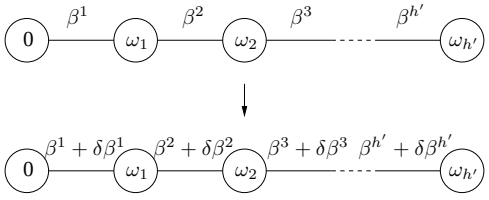

Fig. 10. Load redistribution when $T_f(\omega_{h'}) < T_f(0) = \max\{T_f(\omega)|\omega \in G_{a+b\mathbf{i}}\}$, where $\beta^k > 0$, and β^k is increased by $\delta\beta^k$ ($\delta\beta^k > 0$) for all $1 \le k \le h'$.

Since the load originates from the node 0, there must exist at least one h-hop long path from node 0 to ω_h, and each node along the path receives nonzero load from the previous node, as shown in Fig. 9, where node ω_k ($1 \le k \le h$) along the path is k hops away from node 0. For presentational convenience, we denote the load received by node ω_k as β^k, and $\beta^k > 0$ for all $1 \le k \le h$. We then reduce the finish time of ω_h by redistributing load as follows.

We decrease β^k by $\delta\beta$, and keep $0 < \delta\beta < \beta^k$ for all $1 \le k \le h$, as shown in Fig. 9. After the load redistribution, we have that $T_f'(0) = T_f(0) + \delta\beta T_{cp}$ by constraint (13) in Table 3. As $T_f(0) < \max\{T_f(\omega)|\omega \in G_{a+b\mathbf{i}}\}$, we choose sufficiently small $\delta\beta$, such that $T_f'(0)$ is still earlier than $\max\{T_f(\omega)|\omega \in G_{a+b\mathbf{i}}\}$.

By constraint (12) in Table 3, we have that $\alpha'(\omega_k) = \alpha(\omega_k)$ for $1 \le k < h$, and $\alpha'(\omega_h) = \alpha(\omega_h) - \delta\beta$ after the load redistribution. Since ω_k receives $\delta\beta$ less load from its neighbor, we can keep the start time of ω_k unchanged after the load redistribution for $1 \le k \le h$, i.e., $T_s'(\omega_k) = T_s(\omega_k)$, and constraint (15) of the relaxed MFTM problem is still satisfied. Therefore, we have that $T_f'(\omega_k) = T_f(\omega_k)$ when $1 \le k < h$, and $T_f'(\omega_h) = T_f(\omega_h) - \delta\beta T_{cp} < \max\{T_f(\omega)|\omega \in G_{a+b\mathbf{i}}\}$.

Since the finish time of node 0 is still smaller than $\max\{T_f(\omega)|\omega \in G_{a+b\mathbf{i}}\}$ after the load redistribution, we can further reduce the finish time of the rest nodes with maximum finish time one by one in the network by redistributing load as above, and obtain a smaller $\max\{T_f(\omega)|\omega \in G_{a+b\mathbf{i}}\}$, which contradicts the fact that $\max\{T_f(\omega)|\omega \in G_{a+b\mathbf{i}}\}$ is already minimized.

Case 2: $T_f(0) = \max\{T_f(\omega)|\omega \in G_{a+b\mathbf{i}}\}$.

In this case, there must exist a node with earlier finish time than $T_f(0)$, and it receives nonzero load from at least one of its neighbors. We assume that the node is h' hops away from node 0, and denote it as $\omega_{h'}$, i.e.,

$$T_f(\omega_{h'}) < \max\{T_f(\omega)|\omega \in G_{a+b\mathbf{i}}\}$$

Similarly, we find one h'-hop long path from 0 to $\omega_{h'}$, where every node receives nonzero load from its previous node along the path, as shown in Fig. 10, we denote the load received by node ω_k as β^k, where $1 \le k \le h'$. Next, we firstly reduce the finish time of node 0 without prolonging the maximum finish time by redistributing load as follows.

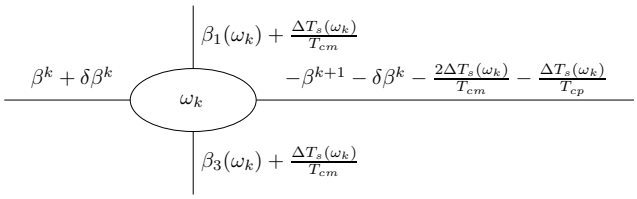

Fig. 11. After the load redistribution, $T_s'(\omega_k) = \Delta T_s(\omega_k) + T_s(\omega_k)$, to avoid delaying the start time of its neighbor 1 and 3, the load sent to these two neighbors, i.e., $\beta_1(\omega_k)$ and $\beta_3(\omega_k)$, are each reduced by $\frac{\Delta T_s(\omega_k)}{T_{cm}}$. Since ω_k receives $\delta\beta^k$ more load from its neighbor 2, it has to send $\delta\beta^k + \frac{2\Delta T_s(\omega_k)}{T_{cm}} + \frac{\Delta T_s(\omega_k)}{T_{cp}}$ more load to its neighbor 0, such that $\alpha'(\omega_k) = \alpha(\omega_k) - \frac{\Delta T_s(\omega_k)}{T_{cp}}$, and its finish time remains the same before and after the load redistribution.

We increase β^k by $\delta\beta^k$, and have that $T_f'(0) = T_f(0) - \delta\beta^1 T_{cp}$. After the load redistribution, we denote $T_s'(\omega_k) - T_s(\omega_k)$ as $\Delta T_s(\omega_k)$, the time when ω_k finishes receiving load from ω_{k-1} is then delayed by $\Delta T_s(\omega_{k-1}) + \delta\beta^k T_{cm}$, where $1 \le k \le h'$. Note that when $k = 1$, $\omega_{k-1} = 0$, and $T_s'(0) = T_s(0) = 0$. We let $T_s'(\omega_k) = T_s(\omega_k) + \Delta T_s(\omega_k)$ such that constraint (15) is still satisfied after the load redistribution. Since $\Delta T_s(\omega_1) = T_s'(0) - T_s(0) + \delta\beta^1 T_{cm} = \delta\beta^1 T_{cm}$ and $\Delta T_s(\omega_k) = \Delta T_s(\omega_{k-1}) + \delta\beta^k T_{cm}$, where $1 < k \le h'$, we have that

$$\Delta T_s(\omega_k) = \sum_{l=1}^{k} \delta\beta^l T_{cm} \tag{16}$$

for $1 \le k \le h'$. To avoid increasing the finish time of ω_k when $1 \le k < h'$, we let

$$\alpha'(\omega_k) = \alpha(\omega_k) - \frac{\Delta T_s(\omega_k)}{T_{cp}}$$

and have that $T_f'(\omega_k) = T_f(\omega_k)$ by constraint (4).

In addition, since each node in Gaussian network has 4 neighbors, ω_k may also send load to another 2 neighbors besides $x_{k+1} + y_{k+1}\mathbf{i}$ when $1 \le k < h'$, as shown in Fig. 11, where we assume that ω_k sends nonzero load to its neighbor 1 and 3, and that $x_{k+1} + y_{k+1}\mathbf{i}$ is its neighbor 0 without loss of generality. To prevent increasing the start time of $n_1(\omega_k)$ and $n_3(\omega_k)$ due to the delayed start time of ω_k, in the worst case, $\beta_1(\omega_k)$ and $\beta_3(\omega_k)$ each have to be decreased by $\frac{\Delta T_s(\omega_k)}{T_{cm}}$ in Fig. 11. Therefore, we have that

$$\delta\beta^{k+1} = \delta\beta^k + \frac{2\Delta T_s(\omega_k)}{T_{cm}} + \frac{\Delta T_s(\omega_k)}{T_{cp}}$$

in the worst case for $1 \le k < h'$.

On the other hand, if ω_k sends no load to $n_1(\omega_k)$ or $n_3(\omega_k)$,

$$\delta\beta^{k+1} = \delta\beta^k + \frac{\Delta T_s(\omega_k)}{T_{cp}}$$

Hence, we have that

$$\delta\beta^k + \frac{\Delta T_s(\omega_k)}{T_{cp}} \le \delta\beta^{k+1} \le \delta\beta^k + \frac{2\Delta T_s(\omega_k)}{T_{cm}} + \frac{\Delta T_s(\omega_k)}{T_{cp}} \tag{17}$$

in general.

The above inequality indicates that $\delta\beta^{l-1} \le \delta\beta^l$ for $1 \le l \le k$, applying Eq. (16), we have that

$$\Delta T_s(\omega_k) \le k\delta\beta^k T_{cm} \tag{18}$$

for $1 \le k \le h'$, which implies

$$\delta\beta^{k+1} \le \delta\beta^k (1 + 2k + \frac{T_{cm}}{T_{cp}} k) \tag{19}$$

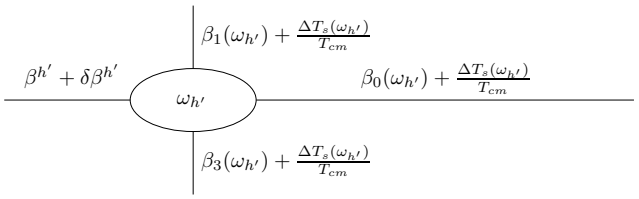

Fig. 12. After the load redistribution, $T'_s(\omega_{h'}) = \Delta T_s(\omega_{h'}) + T_s(\omega_{h'})$, to avoid delaying the start time of its neighbor 0, 1 and 3, the load sent to these 3 neighbors, i.e., $\beta_0(\omega_{h'})$, $\beta_1(\omega_{h'})$ and $\beta_3(\omega_{h'})$, are each reduced by $\frac{\Delta T_s(\omega_{h'})}{T_{cm}}$. Since $\omega_{h'}$ receives $\delta\beta^{h'}$ more load from its neighbor 2, we have that $\alpha'(\omega_{h'}) = \alpha(\omega_{h'}) + \frac{3\Delta T_s(\omega_{h'})}{T_{cm}} + \delta\beta^{h'}$ after the load redistribution.

therefore,

$$\delta\beta^k \leq \delta\beta^1 \prod_{l=0}^{k-1}(1 + 2l + \frac{T_{cm}}{T_{cp}}l) \tag{20}$$

and

$$\Delta T_s(\omega_k) \leq k\delta\beta^1 T_{cm} \prod_{l=0}^{k-1}(1 + 2l + \frac{T_{cm}}{T_{cp}}l) \tag{21}$$

where $1 \leq k \leq h'$.

As for node $\omega_{h'}$, since $T'_s(\omega_{h'}) = T_s(\omega_{h'}) + \Delta T_s(\omega_{h'})$, and $\omega_{h'}$ may send load to its 3 neighbors other than $x_{h'-1} + y_{h'-1}\mathbf{i}$, as shown in Fig. 12, where we assume that $\omega_{h'}$ sends nonzero load to its neighbor 0, 1 and 3, and $x_{h'-1} + y_{h'-1}\mathbf{i}$ is its neighbor 2 without loss of generality. To avoid delaying the start time of $n_0(\omega_{h'})$ $n_1(\omega_{h'})$ and $n_3(\omega_{h'})$, in the worst case, $\beta_0(\omega_{h'})$, $\beta_1(\omega_{h'})$, $\beta_3(\omega_{h'})$ each have to be decreased by $\Delta T_s(\omega_{h'})$. Therefore, we have that

$$\alpha'(\omega_{h'}) = \alpha(\omega_{h'}) + \delta\beta^{h'} + \frac{3\Delta T_s(\omega_{h'})}{T_{cm}}$$

in the worst case, and

$$T'_f(\omega_{h'}) = T_f(\omega_{h'}) + \Delta T_s(\omega_{h'}) + (\delta\beta^{h'} + \frac{3\Delta T_s(\omega_{h'})}{T_{cm}})T_{cp} \tag{22}$$

By Eq. (20), (21) and (22), we have that

$$T'_f(\omega_{h'}) \leq T_f(\omega_{h'}) + \delta\beta^1 T_{cp} \prod_{l=0}^{h'-1}(1 + 2l + \frac{T_{cm}}{T_{cp}}l) + (1 + 3\frac{T_{cp}}{T_{cm}})h'\delta\beta^1 T_{cm} \prod_{l=0}^{h'-1}(1 + 2l + \frac{T_{cm}}{T_{cp}}l) \tag{23}$$

Since $T_f(\omega_{h'}) < \max\{T_f(\omega)|\omega \in G_{a+b\mathbf{i}}\}$ before the load redistribution, we can choose sufficiently small $\delta\beta^1 > 0$, such that $T'_f(\omega_{h'})$ is still earlier than $\max\{T_f(\omega)|\omega \in G_{a+b\mathbf{i}}\}$ after the load redistribution.

In summary, after the load redistribution, we have that

$$T'_f(0) = T_f(0) - \delta\beta^1 T_{cp} < \max\{T_f(\omega)|\omega \in G_{a+b\mathbf{i}}\}$$

$T'_f(\omega_{h'}) < \max\{T_f(\omega)|\omega \in G_{a+b\mathbf{i}}\}$ and $T'_f(\omega_k) = T_f(\omega_k)$ when $1 \leq k < h'$. Since $T'_f(0)$ is now smaller than $\max\{T_f(\omega)|\omega \in G_{a+b\mathbf{i}}\}$, we can reduce all nodes with maximum finish time in the network by the load redistribution in case 1 to obtain a smaller $\max\{T_f(\omega)|\omega \in G_{a+b\mathbf{i}}\}$, which also contradicts that the relaxed MFTM problem is optimized before the load redistribution.

Hence, if $T_s(\omega) > 0$, $T_f(\omega)$ must equal to $T_f(0)$ when the relaxed MFTM problem is optimized. ■

Theorem 1 indicates the following corollary.

Corollary 5: When the relaxed MFTM problem is optimized, $\forall\omega \neq 0,\ T_s(\omega) > 0$.

Proof: We prove the corollary by contradiction, and assume that there exist $\omega \neq 0$, such that $T_s(\omega) = 0$. By Theorem 1, $\forall\omega \neq 0$, $T_f(\omega)$ equals to either $T_f(0)$ or 0, therefore, we can always find a pair of neighboring nodes, say, ω_1 and ω_2, such that $T_f(\omega_1) = T_f(0)$, $T_s(\omega_2) = T_f(\omega_2) = 0$, and ω_1 is either node 0 or closer to node 0 than ω_2. Next, we redistribute load by letting ω_1 send a sufficiently small fraction of load to ω_2, such that the finish time of ω_2 is nonzero, but smaller than $\max\{T_f(\omega)|\omega \in G_{a+b\mathbf{i}}\}$. In addition, ω_1 processes less load after the load redistribution, and should finish processing load earlier. Therefore, the load redistribution will not result in a greater maximum finish time of all nodes in $G_{a+b\mathbf{i}}$, but now ω_2 receives nonzero load from ω_1, and its finish time is smaller than $\max\{T_f(\omega)|\omega \in G_{a+b\mathbf{i}}\}$, which contradicts Theorem 1. ■

Theorem 1 and Corollary 5 in conjunction prove Theorem 2.

Theorem 2: When the relaxed MFTM problem is optimized, $T_f(\omega)$ must be equal for all $\omega \in G_{a+b\mathbf{i}}$.

With Theorem 2, we show that the MFTM problem and relaxed MFTM problem share identical optimal solution in the next theorem.

Theorem 3: U^* is the optimal solution of the relaxed MFTM problem.

Proof: Since constraint (15) is relaxed from constraint (6), U^* is feasible for the relaxed MFTM problem. Therefore, the MFTM problem can not have a better optimal solution than the relaxed MFTM problem.

We then prove the theorem by showing that the optimal solution of the relaxed MFTM problem is also a feasible solution of the MFTM problem, which means that the relaxed MFTM problem has no better optimal solution than the MFTM problem either, and these two problems must have equal optimal solution.

Assume that the optimal solution of the relaxed MFTM problem is not feasible for the MFTM problem, there must exist at least one node, say, ω, such that

$$T_s(\omega) > \max\{T_s(n_j(\omega)) + \beta_j(\omega)T_{cm}|\beta_j(\omega) > 0\}$$

Therefore, we can assign ω an earlier start time, denoted as $T'_s(\omega)$, and

$$T'_s(\omega) = \max\{T_s(n_j(\omega)) + \beta_j(\omega)T_{cm}|\beta_j(\omega) > 0\}$$

The new solution is still optimal for the relaxed MFTM problem, but the finish time of ω is now earlier than $\max\{T_f(\omega)|\omega \in G_{a+b\mathbf{i}}\}$, which contradicts Theorem 2, i.e., all nodes should have equal finish time when the relaxed MFTM problem is optimized. Hence, the assumption is false, and the optimal solution of the relaxed MFTM problem is feasible for the MFTM problem. ■

With the Theorem 2 and 3, we further transfer the relaxed MFTM problem to the ***finish time minimization (FTM)*** problem, of which the optimal solution is also U^*, in the next subsection, and propose an optimal algorithm for the FTM problem.

C. Finish Time Minimization (FTM) Problem

Since all nodes have to finish processing load simultaneously by Theorem 2 when the relaxed MFTM problem is optimized, constraint (4) can be replaced by

$$T_f = T_s(\omega) + \alpha(\omega)T_{cp}$$

and the object of the relaxed MFTM problem is to minimize T_f. In addition, all $\beta_j(\omega)$s satisfying constraint (15) must be positive, thus belong to U_β^+. These $\beta_j(\omega)$s consist of a subset of U_β^+, which we denote as S_β^+. Therefore, if we can determine S_β^+ in the optimal solution of the MFTM problem, all constraints and the object function become linear, indicating an optimal solution by linear programming. By the above analysis, we propose the ***finish time minimization (FTM)*** problem in Table 4.

TABLE 4
FINISH TIME MINIMIZATION (FTM) PROBLEM FORMULATION FOR DIVISIBLE LOAD SCHEDULING IN A GAUSSIAN NETWORK

Minimize: $T_f(S_\beta^+)$

Subject to:

$$T_f(S_\beta^+) = T_s(\omega) + \alpha(\omega)T_{cp} \quad (24)$$
$$T_s(0) = 0 \quad (25)$$
$$T_s(\omega) \geq T_s(n_j(\omega)) + \beta_j(\omega)T_{cm}, \text{ if } \beta_j(\omega) \in S_\beta^+ \quad (26)$$
$$\beta_j(\omega) \geq 0, \text{ if } \beta_j(\omega) \in U_\beta^+ \quad (27)$$
$$\beta_j(\omega) \leq 0, \text{ if } \beta_j(\omega) \in U_\beta^- \quad (28)$$
$$\beta_j(\omega) = 0, \text{ if } \beta_j(\omega) \in U_\beta^0 \cup (U_\beta^+ - S_\beta^+) \quad (29)$$
$$\beta_j(\omega) = -\beta_k(n_j(\omega)), \text{ if } k = \text{mod}_4(j+2) \quad (30)$$
$$\alpha(\omega) = \sum_{j=0}^{3} \beta_j(\omega), \text{ if } \omega \neq 0 \quad (31)$$
$$\alpha(0) = 1 + \sum_{j=0}^{3} \beta_j(0) \quad (32)$$
$$\alpha(\omega) \geq 0 \quad (33)$$
$$S_\beta^+ \subseteq U_\beta^+ \quad (34)$$

In the FTM problem, S_β^+ can be any subset of U_β^+, and if $\beta_j(\omega) \in U_\beta^+ - S_\beta^+$, we let $\beta_j(\omega) = 0$, as stated by constraint (29). All nodes share the same finish time, which is a function of S_β^+, and is denoted as $T_f(S_\beta^+)$. The object of the FTM problem is to minimize $T_f(S_\beta^+)$, and we have the following theorem on the optimal solution of the FTM problem.

Theorem 4: U^* is the optimal solution of the FTM problem.

Proof: Since all nodes have equal finish time in the optimal solution of the relaxed MFTM problem, and that S_β^+ can be any subset of U_β^+, U^*, the optimal solution of the relaxed MFTM problem, must be feasible for the FTM problem, meaning that the relaxed MFTM problem has no better optimal solution than the FTM problem. On the other hand, given a feasible solution of the FTM problem, if $\beta_j(\omega) > 0$, we have that $\beta_j(\omega) \in S_\beta^+$, and constraint (26) is satisfied, therefore, any feasible solution of the FTM problem should also be feasible for the relaxed MFTM problem, therefore, the FTM problem has no better optimal solution than the relaxed MFTM problem either, and these two problems must have equal optimal solution. ■

Lemma 3: Define

$$S_\beta^* = \{\beta_j(\omega) | \beta_j(\omega) > 0, \beta_j(\omega) \in U^*\}$$

when the FTM problem is optimized, $S_\beta^+ = S_\beta^*$.

Proof: Since if $\beta_j(\omega) \geq 0$, $\beta_j(\omega) \in U_\beta^+$, we have that $S_\beta^* \subseteq U_\beta^+$. In addition, by Theorem 4, U^* is the optimal solution of the FTM problem, thus, if $S_\beta^+ = S_\beta^*$, constraint (26) and (29) will both be satisfied. ■

Next, we propose an optimal algorithm based on linear programming, denoted as *LP-based* algorithm, for the FTM problem by Lemma 3.

We notice that constraint (24)-(33) are all linear, therefore, for a given S_β^+, to minimize $T_f(S_\beta^+)$ becomes a linear optimization problem, which we denote as $LP(S_\beta^+)$. That is, for a given subset of U_β^+, S_β^+, $LP(S_\beta^+)$ is a linear optimization problem, which minimizes $T_f(S_\beta^+)$ under constraint (24)-(33) in Table 4. By Lemma 3, the FTM problem will be optimized when $S_\beta^+ = S_\beta^*$. Since U_β^+ has $2^{|U_\beta^+|}$ subsets, we can try each of them, and solve $2^{|U_\beta^+|}$ linear optimization problems to find out the optimal solution.

It is worth mentioning that S_β^* has some features, which are useful in determining whether $T_f(S_\beta^+)$ is the minimum finish time. In other words, if a given S_β^+ does not have the same features as S_β^*, we will not find the optimal solution of the FTM problem by solving $LP(S_\beta^+)$. These features are listed as follows.

- $\forall \omega \in G_{a+b\mathbf{i}}$ and $\omega \neq 0$, $\exists j \in \{0,1,2,3\}$ such that $\beta_j(\omega) \in S_\beta^*$.
- $\beta_j(\omega_1) \in S_\beta^*$ if and only if $\beta_k(\omega_2) \in S_\beta^*$, where $\omega_2 = \omega_1 \cdot \mathbf{i}$, $k = \text{mod}_4(j+1)$.
- When a and b are both even, if $\beta_j(\frac{a}{2} + \frac{b}{2}\mathbf{i})$ (or $\beta_j(\frac{b}{2} - \frac{a}{2}\mathbf{i})$) is in S_β^*, then $\beta_k(\frac{a}{2} + \frac{b}{2}\mathbf{i})$ (or $\beta_k(\frac{b}{2} - \frac{a}{2}\mathbf{i})$) is also in S_β^*, where $k = \text{mod}_4(j+2)$.
- If $G_{a+b\mathbf{i}}$ is an optimal Gaussian network, $\beta_0(x+y\mathbf{i})$, $\beta_0(x-y\mathbf{i})$, $\beta_1(-y+x\mathbf{i})$, $\beta_1(y+x\mathbf{i})$, $\beta_2(-x-y\mathbf{i})$, $\beta_2(-x+y\mathbf{i})$, $\beta_3(y-x\mathbf{i})$ and $\beta_3(-y-x\mathbf{i})$ are all in S_β^* if one of them is in S_β^*.
- When $a+b$ is even, $\beta_0(-\frac{a+b}{2} + \frac{a-b}{2}\mathbf{i})$, $\beta_1(-\frac{a+b}{2} + \frac{a-b}{2}\mathbf{i})$, $\beta_2(-\frac{a+b}{2} + \frac{a-b}{2}\mathbf{i})$ and $\beta_3(-\frac{a+b}{2} + \frac{a-b}{2}\mathbf{i})$ are all in S_β^*.

The first feature originates from the fact that all nodes should have equal finish time in U^* by $T_f(\omega)$, therefore, every node except for the source node in the network should receive load from at least one of its neighbors.

By Lemma 2, $\beta_j(\omega_1) = \beta_k(\omega_2)$ when $\omega_2 = \omega_1 \cdot \mathbf{i}$ and $k = \text{mod}_4(j+1)$ in U^*, thus, they are in or not in S_β^* concurrently. Similarly, S_β^* has the next two features by Corollary 2 and 3.

Since that when $a+b$ is even, $\beta_0(-\frac{a+b}{2}+\frac{a-b}{2}\mathbf{i})$, $\beta_1(-\frac{a+b}{2}+\frac{a-b}{2}\mathbf{i})$, $\beta_2(-\frac{a+b}{2}+\frac{a-b}{2}\mathbf{i})$ and $\beta_3(-\frac{a+b}{2}+\frac{a-b}{2}\mathbf{i})$ are equal in U^* by Corollary 4, and at least one of them is in S_β^* according to the first feature of S_β^*, therefore, they are all in S_β^*.

With these features of S_β^*, a high-level description of the LP-based algorithm is given in Table 5.

Since there are total $2(a^2+b^2)$ links in Gaussian network $G_{a+b\mathbf{i}}$, we have $|U_\beta^+| \leq 2(a^2+b^2)$, which means that the number of subsets of U_β^+, i.e., $2^{|U_\beta^+|}$, might grow exponentially with network size. Though the features of S_β^* can help exclude some subsets of U_β^+, the number of linear optimization problems that the LP-based algorithm in Table 5 needs to solve is expected to increase quickly as the network size grows, and we will analyze its time complexity in the next subsection.

Considering the high time complexity of the LP-based algorithm, we propose a heuristic algorithm for the FTM problem, which solves only one linear optimization problem. The idea of the heuristic algorithm is that every node in the network receives load from

TABLE 5
HIGH-LEVEL DESCRIPTION OF THE LP-BASED ALGORITHM FOR THE FTM PROBLEM

for each $S_\beta^+ \subseteq U_\beta^+$
  if S_β^+ has the same features as S_β^*
    solve $LP(S_\beta^+)$, which minimizes $T_f(S_\beta^+)$
    under constraint (24)-(33) in Table 4;
  end if;
end for;
The FTM problem and $LP(S_\beta^+)$ with minimum object function value have equal optimal solution.
End

all its neighbors that are closer to node 0. Hence, in the heuristic algorithm, we let $S_\beta^+ = U_\beta^+$, and solve the linear optimization problem $LP(U_\beta^+)$. The optimal solution of $LP(U_\beta^+)$ is then used as the solution of the heuristic algorithm. As will be seen in Section V, the performance of our proposed heuristic algorithm is extremely close, and even equal in many cases, to the LP-based algorithm in terms of finish time.

D. Time Complexity Analysis

In this subsection, we analyze the time complexity of the LP-based algorithm and the heuristic algorithm relative to the network size. Since the LP-based algorithm solves numerous linear optimization problems $LP(S_\beta^+)$, where $S_\beta^+ \subseteq U_\beta^+$, and the heuristic algorithm solves only $LP(U_\beta^+)$, we begin with the analysis of the time complexity of solving $LP(S_\beta^+)$.

The standard form of the linear optimization problem is to maximize $\mathbf{c}^T\mathbf{x}$ ($\mathbf{c}, \mathbf{x} \in \mathbf{R}^n$) over all vectors $\mathbf{x}$ such that $\mathbf{Ax} = \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$. In 1979, Khachiyan showed that a linear optimization problem can be solved in polynomial time relative to the length of the binary encoding of the input, denoted as L [33]. In other words, L is the number of bits encoding **A**, **b** and **c**.

To convert $LP(S_\beta^+)$ to the above standard form, we eliminate all $\beta_j(\omega)$s that are zero by constraint (29), substitute $\beta_j(\omega) \in U_\beta^-$ with $-\beta_k(n_j(\omega))$, where $\beta_k(n_j(\omega)) \in U_\beta^+,\ k = \text{mod}_4(j+2)$, and add nonnegative slack variables, denoted as $T_j(\omega)$, in constraint (26) to transform the inequality to equality, i.e.,

$$T_s(\omega) = T_s(n_j(\omega)) + \beta_j(\omega)T_{cm} + T_j(\omega)$$

After these operations, there are $2(a^2+b^2+|S_\beta^+|)+1$ nonnegative variables and $2(a^2+b^2)+|S_\beta^+|+1$ equalities in $LP(S_\beta^+)$.

By the features of S_β^* listed in the previous subsection, we have the following equation regarding the cardinality of S_β^+.

$$|S_\beta^+| \geq \begin{cases} a^2+b^2+2,\ a,b \text{ are both odd} \\ a^2+b^2+4,\ a,b \text{ are both even} \\ a^2+b^2-1,\ a+b \text{ is odd, and } a > b+1 \\ a^2+b^2-1+4\lfloor\frac{b}{2}\rfloor,\ a = b+1 \end{cases} \tag{35}$$

In addition,

$$|U_\beta^+| = \begin{cases} 2(a^2+b^2),\ a+b \text{ is even} \\ 2(a^2+b^2-2a+1),\ a+b \text{ is odd} \end{cases} \tag{36}$$

and $|S_\beta^+| \leq |U_\beta^+|$, indicating that L in $LP(S_\beta^+)$ is polynomial to the network size, the time complexity of solving $LP(S_\beta^+)$ is thus also polynomial to the network size by Khachiyan's result. Note that the dependencies among variables by Lemma 2 and Corollary 2-4 can reduce the number of variables and equalities by a factor of 4 (or 8 when the Gaussian network is optimal), which can help improve the efficiency of solving $LP(S_\beta^+)$ in practice despite the same time complexity in theory.

Next, we discuss the number of linear optimization problems that the LP-based algorithm solves. As $\beta_j(\omega \cdot \mathbf{i}^j)$, where $j = 0, 1, 2$ and 3, are in S_β^+ concurrently by the second feature of S_β^*, and when $a = b+1$, $\beta_0(x+y\mathbf{i})$, $\beta_0(x-y\mathbf{i})$, $\beta_1(-y+x\mathbf{i})$, $\beta_1(y+x\mathbf{i})$, $\beta_2(-x-y\mathbf{i})$, $\beta_2(-x+y\mathbf{i})$, $\beta_3(y-x\mathbf{i})$ and $\beta_3(-y-x\mathbf{i})$ are in S_β^+ concurrently according to the forth feature of S_β^*, we can construct at least $2^{\frac{|U_\beta^+|-\min\{|S_\beta^+|\}}{4}}$ subsets of U_β^+ when $a \neq b+1$, and no less than $2^{\frac{|U_\beta^+|-\min\{|S_\beta^+|\}}{8}}$ subsets of U_β^+ when $a = b+1$, which share all features of S_β^*. Besides, U_β^+ has $2^{|U_\beta^+|}$ subsets in total, therefore, the number of linear optimization problems that the LP-based algorithm in Table 5 needs to solve increases exponentially with the network size, and the LP-based algorithm has exponential time complexity relative to network size.

It is worth pointing out that the ellipsoid algorithm, which was used by Khachiyan to prove his result, is not useful for solving linear optimization problems in practice. On the other hand, the widely used simplex algorithm for solving linear optimization problems is very efficient in practice despite that there exist constructed examples which require exponential time by the simplex algorithm. Hence, we also use the simplex algorithm to solve $LP(S_\beta^+)$ in our paper.

As mentioned in Section I, our proposed MFTM problem formulation of divisible load scheduling in Gaussian network can be readily extended to mesh and torus, which will be discussed in the next section.

IV. EXTENSION OF MFTM PROBLEM FORMULATION TO MESH AND 2D-TORUS

In this section, we extend the MFTM problem formulation to mesh and torus.

Since the coordinates of nodes are Gaussian integers in the MFTM problem, we place nodes of $a \times b$ mesh and $a \times b$ torus in the complex plane as well, and use Gaussian integers in an $a \times b$ rectangle with 0, $a-1$, $(b-1)\mathbf{i}$ and $(a-1)+(b-1)\mathbf{i}$ as vertices to represent nodes in the network. Fig. 13 is an example of node placement of 5×5 mesh and 5×5 torus in the complex plane, where Gaussian integers in a 5×5 square represent nodes in the network.

In the mesh, ω is adjacent to $\omega + \mathbf{i}^j$, where $j = 0, 1, 2$ and 3, if $\omega + \mathbf{i}^j$ is in the mesh, and we say that $\omega + \mathbf{i}^j$ is neighbor j of ω, i.e.,

$$n_j(\omega) = \omega + \mathbf{i}^j,\ \text{if } n_j(\omega) \text{ is in the mesh} \tag{37}$$

In the torus, node $x + y\mathbf{i}$ is adjacent to $\text{mod}_a(x \pm 1) + y\mathbf{i}$, $x + \text{mod}_b(y \pm 1)\mathbf{i}$, and we let

$$n_j(x+y\mathbf{i}) = \begin{cases} \text{mod}_a(x+1)+y\mathbf{i} & \text{if } j = 0 \\ x+\text{mod}_b(y+1)\mathbf{i} & \text{if } j = 1 \\ \text{mod}_a(x-1)+y\mathbf{i} & \text{if } j = 2 \\ x+\text{mod}_b(y-1)\mathbf{i} & \text{if } j = 3 \end{cases} \tag{38}$$

By Eq. (37) and (38), given two nodes, say, $\omega_1 = x_1 + y_1\mathbf{i}$ and $\omega_2 = x_2 + y_2\mathbf{i}$, we have that $\mathbf{D}(\omega_1, \omega_2)$, i.e., the distance between ω_1 and ω_2, is $|x_2 - x_1| + |y_2 - y_1|$ and $\min\{\text{mod}_a(x_2 - x_1), \text{mod}_a(x_1 -$

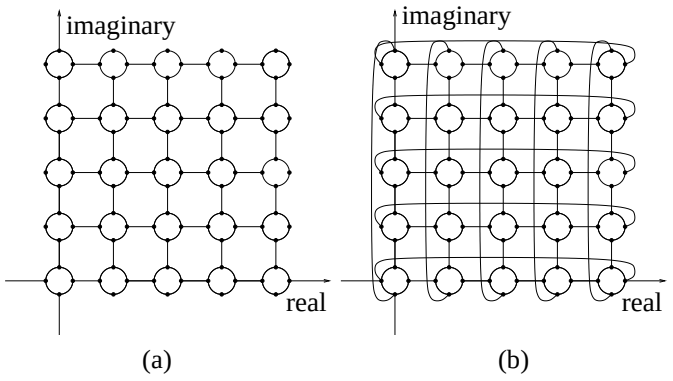

Fig. 13. Node placement of 5×5 mesh and 5×5 torus in a square with 0, 4, 4**i** and $4 + 4$**i** as the vertices in the complex plane.

$x_2)\} + \min\{\text{mod}_b(y_2 - y_1), \text{mod}_b(y_1 - y_2)\}$ in mesh and torus, respectively. Suppose that the load originates from node ω_0, ω can only receive load from its neighbors that are closer to ω_0 than itself, meaning that $\beta_j(\omega) \geq 0$ if $\mathbf{D}(\omega_0, \omega) > \mathbf{D}(\omega_0, n_j(\omega))$. In addition, according to Eq. (37) and (38), ω is still neighbor $\text{mod}_4(j+2)$ of $n_j(\omega)$ in mesh and torus, hence, $\beta_j(\omega) = -\beta_k(n_j(\omega))$, where $k = \text{mod}_4(j+2)$. Note that since nodes on the boundary of the $a \times b$ rectangle in the mesh have less than 4 neighbors, and we let $\beta_j(\omega) = 0$ if ω does not have neighbor j in the mesh. Also, ω can start sending out and processing load after it finishes receiving load from all its neighbors.

To formulate the divisible load scheduling problem in mesh and torus as the MFTM problem, U_β^+, U_β^- and U_β^0 are defined as follows. In the mesh,

$$U_\beta^+ = \{\beta_j(\omega) | \mathbf{D}(\omega_0, \omega) > \mathbf{D}(\omega_0, n_j(\omega))\}$$
$$U_\beta^- = \{\beta_j(\omega) | \mathbf{D}(\omega_0, \omega) < \mathbf{D}(\omega_0, n_j(\omega))\}$$
$$U_\beta^0 = \{\beta_j(\omega) | \omega \text{ does not have neighbor } j\}$$

As for the torus,

$$U_\beta^+ = \{\beta_j(\omega) | \mathbf{D}(\omega_0, \omega) > \mathbf{D}(\omega_0, n_j(\omega))\}$$
$$U_\beta^- = \{\beta_j(\omega) | \mathbf{D}(\omega_0, \omega) < \mathbf{D}(\omega_0, n_j(\omega))\}$$
$$U_\beta^0 = \{\beta_j(\omega) | \mathbf{D}(\omega_0, \omega) = \mathbf{D}(\omega_0, n_j(\omega))\}$$

By defining U_β^+, U_β^- and U_β^0 as above, we still have that $\beta_j(\omega) \geq 0$, $\beta_j(\omega) \leq 0$ and $\beta_j(\omega) = 0$ for $\beta_j(\omega)$ in U_β^+, U_β^- and U_β^0, respectively. The divisible load scheduling in mesh and torus is then formulated as the MFTM problem in Table 3 as well, which can be transformed into the FTM problem in Table 4 by Theorem 2, 3 and 4, and we can obtain the optimal solution of the corresponding FTM problem by solving $2^{|U_\beta^+|}$ linear optimization problems. Also, we can let $S_\beta^+ = U_\beta^+$, and use the solution of $LP(U_\beta^+)$ as the suboptimal solution for the FTM problem. Hence, the heuristic algorithm proposed in the previous section applies to mesh and torus networks as well.

Next, we will compare the performance of our proposed heuristic algorithm with the LP-based algorithm, and the previously proposed dimensional algorithm and phase algorithm.

V. Performance Evaluation

In this section, we compare the performance of our proposed heuristic algorithm with the LP-based algorithm, and the previously proposed *dimensional algorithm* [13], pipeline algorithms [14] and *phase algorithm* [16] in the Gaussian network, mesh and torus with respect to different network sizes. As mentioned in Section III, we rely on simplex algorithm to solve $LP(S_\beta^+)$. To evaluate the efficiency of the simplex algorithm, we count the number of iterations by the simplex method to solve $LP(U_\beta^+)$ in the heuristic algorithm.

In [13], the dimensional algorithm is proposed for N-dimensional mesh and torus, in which an $a_1 \times a_2 \times ... \times a_N$ mesh (or torus) is considered as a_N linearly connected $a_1 \times a_2 \times ... \times a_{N-1}$ meshes (or tori), and a single node is regarded as the 0-dimensional mesh (or torus). Since a daisy chain network, i.e., linearly connected nodes, of processors is equal to a single processor with faster processing speed [6], an N-dimensional mesh (or torus) can recursively be equivalent to a single processor under the dimensional algorithm. Based on the dimensional algorithm, two pipeline algorithms, named as pipeline 1 algorithm and pipeline 2 algorithm, respectively, are proposed in [14] to reduce the overhead of distributing load in one dimension. Pipeline 1 algorithm allows the node to start transmitting load before finishing receiving load, but all nodes except for the source node in the same dimension must start processing load after load distribution completes in the dimension. In pipeline 2 algorithm, nodes are allowed to start both processing and transmitting load before finishing receiving load.

The phase algorithm is proposed for divisible load scheduling in torus networks in [16]. In the phase algorithm, the circuit switching mechanism is adopted such that a node can send load to remote nodes directly as long as no link along the routing path is occupied, which is quite different from the other algorithms in our comparison, which allow nodes to send load only to their neighbors. For example, the phase algorithm allows load being sent from node 0 to node 2 directly in the 5×5 torus in Fig. 13 (b) along the link from node 0 to node 1 and the link from node 1 to node 2, which bypasses node 1. The load is distributed to nodes in the network in several phases. In phase 0, the node where load originates starts load distribution, and sends load to 4 nodes since each node has 4 ports in the torus. Next, in phase 1, 5 nodes hold load, and each of them distributes load to another 4 nodes in the network. Therefore, in phase N, load will be distributed to 5^N nodes in total. To prevent link contention in each phase, a node may have to send load to remote nodes, which will incur a long startup time, for simplicity, we set the startup time to be zero in our comparison. The closed form solutions of load distribution by the above 4 algorithms are given in the corresponding references.

In the performance comparison, we set T_{cp} as 1, and increase T_{cm} from 0.01 to 10, corresponding to computation-intensive load and communication-intensive load, and the incremental steps are 0.01, 0.1 and 1 in intervals [0.01,0.1], [0.1,1] and [1,10], respectively. The performance of the compared algorithms is evaluated in terms of speedup, which is the ratio between the finish time of total load by a single node and that by the whole network.

A. Comparison in Gaussian Networks

In this subsection, we compare the performance of the heuristic algorithm with the LP-based algorithm in Gaussian networks, and adopt three optimal Gaussian networks, $G_{4+3\mathbf{i}}$, $G_{5+4\mathbf{i}}$ and $G_{6+5\mathbf{i}}$, such that we can reduce the computation complexity, especially when adopting the LP-based algorithm, by taking advantage of the network symmetry to the maximum extent.

The comparison results are plotted in Fig. 14, from which we can see that our proposed heuristic algorithm has extremely close,

and almost equal, performance to the LP-based algorithm under all network sizes. The underlying reason is that all the links in the network are assumed to be active, i.e., used to transmit load, in the heuristic algorithm, though the assumption might not be true in the optimal load distribution, taking advantage of all links in the network to transmit load is an efficient load distribution scheme. Considering that the heuristic algorithm solves only one linear optimization problem, and has much lower time complexity than the LP-based algorithm, the performance of our proposed heuristic algorithm is quite satisfactory. We also observe that larger network size and smaller T_{cm} results in higher speedup under both algorithms. This is because that larger network provides greater computing power and smaller T_{cm} introduces less overhead in load distribution. On the other hand, load distribution becomes less economical under larger T_{cm}, and we can see that larger network size brings about little speedup improvement when $T_{cm} \geq 0.2$, which will also be observed in mesh and torus under a variety of divisible load scheduling algorithms.

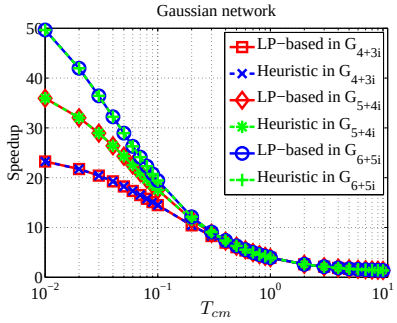

Fig. 14. Speedup comparison between LP-based algorithm and heuristic algorithm with respect to different network sizes and inverse data rates in Gaussian networks.

B. Comparison in Meshes and Tori

In this subsection, we evaluate the performance of our proposed heuristic algorithm in meshes and tori with respect to the LP-based algorithm, heuristic algorithm, dimensional algorithm, pipeline 1 algorithm, pipeline 2 algorithm and phase algorithm.

As in the previous subsection, for the purpose of reducing computation complexity from the network symmetry, we use three square meshes and tori in our comparison, they are 5×5, 7×7 and 9×9 meshes and tori. Since the nodes are asymmetric in the mesh, and the performance of mesh in scheduling divisible load is related to the position of ω_0, i.e., the node where the load originates. Generally speaking, the mesh has better performance in scheduling divisible load when ω_0 is closer to the geometric center, and we let ω_0 be the geometric centers of the selected meshes in our comparison, such that we can observe the best performance of these meshes under given algorithms. The geometric centers of the 5×5, 7×7 and 9×9 meshes are $2+2\mathbf{i}$, $3+3\mathbf{i}$ and $4+4\mathbf{i}$, respectively. As nodes in a torus are symmetric, the divisible load scheduling is independent of the position of ω_0, nevertheless, we still choose $2+2\mathbf{i}$, $3+3\mathbf{i}$ and $4+4\mathbf{i}$ as ω_0 in the 5×5, 7×7 and 9×9 tori, respectively.

The comparison results among these algorithms in the meshes are in Fig. 15. We can see that the heuristic algorithm still have almost equal performance to the LP-based algorithm in the mesh, and that our proposed heuristic algorithm achieves much higher speedup than the dimensional algorithm, and the gap between these two algorithms broadens as the T_{cm} increases. When $0.01 < T_{cm} < 0.2$, our proposed heuristic algorithm increases the speedup by about 12% in the 5×5 mesh, and the speedup improvement reaches around 25% and over 30% in the 7×7 and 9×9 meshes, respectively. The reason is that the dimensional algorithm fails to efficiently use all the links in the network to distribute load. For example, since node 0, 1, 2, 3 and 4 equal to a single process in the 5×5 mesh by the dimensional algorithm, node 1 can receive load only from node 2, while node 1 can receive load from node 2 and $1+\mathbf{i}$ in the heuristic algorithm, which shortens the load transmission time. As the network size grows, the network size grows, and T_{cm} increases, which brings about higher overhead when distributing load, the inefficiency of the dimensional algorithm in utilizing links deteriorates, and the advantage of the heuristic algorithm becomes greater. Though the pipeline 1 algorithm and the pipeline 2 algorithm can shorten load distribution time in one dimension, they still suffer inefficient utilization of links inherited from the dimensional algorithm, and even though nodes can transmit and receive load concurrently in the pipeline 1 algorithm, it never outperforms the heuristic algorithm. Allowing nodes to process and transmit load while receiving load provides pipeline 2 algorithm great advantages over the heuristic algorithm, nevertheless, it is only slightly better than the heuristic algorithm. Furthermore, as the pipeline schemes adopted by pipeline 1 algorithm and pipeline 2 algorithm are independent of divisible load scheduling algorithms, we can integrate them in the heuristic algorithm as well to increase its performance. Under large T_{cm}, distributing load to remote nodes becomes uneconomical, the majority of load is thus processed by the source node and a few nearby nodes, and the speedup is almost independent of the scheduling algorithms.

Fig. 16 plots the comparison results in the tori. Note that since the 5×5 torus can adopt the phase algorithm, we also take it into consideration in our comparison, as shown in Fig. 16 (a). The phase algorithm can use all ports of a node to distribute load in each phase, therefore, it achieves better performance than the other algorithms. However, as mentioned above, only torus with 5^N nodes can adopt the phase algorithm, which restricts its application. Moreover, the phase algorithm may result in long startup time when sending load to remote nodes, which is ignored in our comparison, while the other algorithms will not since nodes only send load to neighbors in these two algorithms. Therefore, the advantages of the phase algorithm over the other algorithms will shrink in real world application when taking the potential long startup time into consideration, which might even render it disadvantageous compared with the other algorithms.

Finally, we notice that mesh and torus of equal network size achieve identical performance when adopting the same algorithm. This is because that when the load originates from the geometric center of the mesh, a given scheduling algorithm results in identical load distribution in the mesh and torus. For example, when adopting the LP-based algorithm and the heuristic algorithm, these two networks have equal U_β^+, U_β^- and U_β^0, meaning that the MFTM problems for them have identical optimal solution, and the suboptimal solutions from the heuristic algorithm will be the same as well.

C. Efficiency of the Simplex Algorithm

As mentioned previously, we rely on the simplex algorithm to solve the $LP(S_\beta^+)$ in the LP-based algorithm and heuristic algorithm. In this subsection, we evaluate the efficiency of the

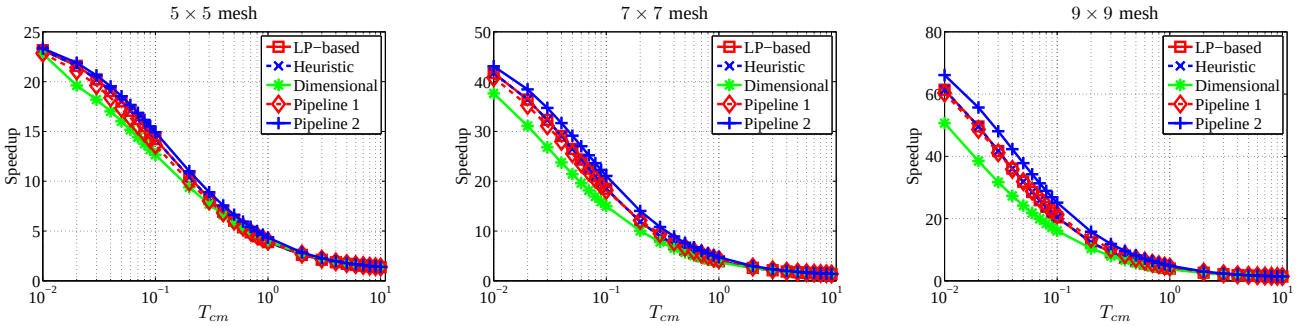

Fig. 15. Speedup comparison among LP-based algorithm, heuristic algorithm, dimensional algorithm, pipeline 1 algorithm and pipeline 2 algorithm with respect to different network sizes and inverse data rates in meshes. (a) 5×5 mesh. (b) 7×7 mesh. (c) 9×9 mesh.

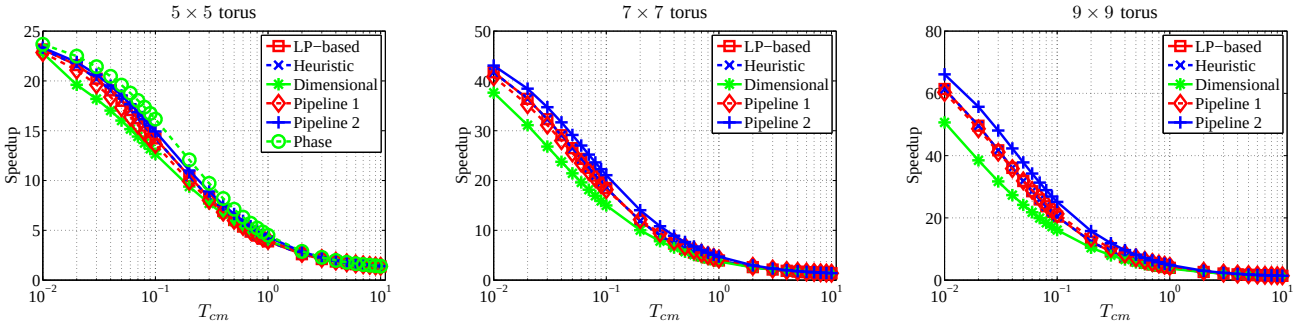

Fig. 16. Speedup comparison among LP-based algorithm, heuristic algorithm, dimensional algorithm, pipeline 1 algorithm, pipeline 2 algorithm and phase algorithm with respect to different network sizes and inverse data rates in tori. (a) 5×5 torus. (b) 7×7 torus. (c) 9×9 torus.

simplex algorithm in terms of the number of iterations taken by the simplex algorithm to solve $LP(U_\beta^+)$ in the heuristic algorithm, as shown in Fig. 17. Note that $a \times a$ mesh and $a \times a$ torus share an identical suboptimal solution by the heuristic algorithm in our simulation, therefore, the number of iterations by the simplex algorithm are equal in these two networks, and we use the line with legend $a \times a$ in Fig. 17 to represent both networks. From Fig. 17, we can see that the number of iterations is small with respect to the corresponding network size, and relatively stable as T_{cm} increases from 0.01 to 10, which is a positive evidence indicating a satisfactory efficiency of the simplex algorithm in solving $LP(U_\beta^+)$. Take the image processing problem of converting an RGB color map to an HSV color map, a typical divisible load task, in the 9×9 mesh (or torus) for example. If each processor in the network has equal computing power as a single core of the Intel Core 2 Quad Core Q9400, it takes roughly 0.5 μs to convert a pixel by such a processor according to our test. Suppose an 8-byte pixel size is used, which is the same as the IKONOS satellite image [34], and that the inter-processor bandwidth is 1 GB/s [35], then to transmit a pixel would be 8 ns. In this case, $\frac{T_{cm}}{T_{cp}} = 0.016$, and the speedup of the network is 53 by running the heuristic algorithm, which takes 0.0395 seconds in a single processor. Therefore, the runtime of the heuristic algorithm equals the processing time of about 4 million pixels, the size of which is 32 MB, by the network. Considering that the RAM size of a PC can be easily scaled up to tens and even hundreds of GB nowadays, the overhead of running the heuristic algorithm should be negligible when the size of processed images is comparable to the RAM size.

In summation, our proposed heuristic algorithm has a significantly lower time complexity than the LP-based algorithm, but

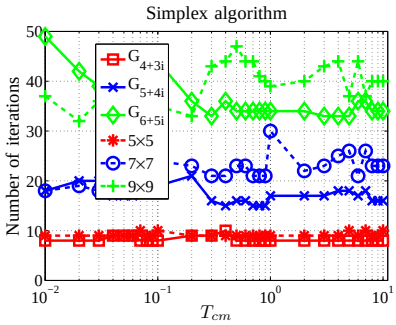

Fig. 17. Number of iterations by the simplex method in solving $LP(U_\beta^+)$ with respect to different network sizes and topologies.

maintains almost equally good performance compared to the latter one. In addition, the heuristic algorithm significantly outperforms the dimensional algorithm, and is much more widely applicable than the phase algorithm.

VI. Conclusions

In this paper, we have formulated the divisible load scheduling in mesh, torus and Gaussian network as the maximum finish time minimization (MFTM) problem, which minimizes the maximum finish time of all nodes in the network. By linearly relaxing the constraints of MFTM problem, we obtain the relaxed MFTM problem, which is proved to have equal optimal solution as the MFTM problem. We showed that the all nodes should have equal finish time when the relaxed MFTM problem is optimized, and further transform it into the finish time minimization (FTM) problem. The FTM problem and MFTM problem have equal optimal solutions

as well, and we propose an optimal algorithm based on linear programming, denoted as the LP-based algorithm, for the FTM problem. Considering the high time complexity of the LP-based algorithm, a heuristic algorithm is also proposed.

Our simulation results demonstrate that the heuristic algorithm achieve extremely close, and even equal, performance to the LP-based algorithm in terms of speedup in all the studied networks. In addition, our proposed heuristic algorithm outperforms the previously proposed dimensional algorithm, and has much wider application range than the phase algorithm. Hence, the MFTM problem formulation is efficient for divisible load scheduling in the Gaussian network, mesh and torus, and our proposed heuristic algorithm greatly improves the performance of mesh and torus in scheduling divisible load, which is a promising scheduling algorithm for real world application.

REFERENCES

[1] G. N. Iyer, B. Veeravalli and S. G. Krishnamoorthy, "On Handling Large-Scale Polynomial Multiplications in Compute Cloud Environments using Divisible Load Paradigm," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, no. 1, pp. 820-831, Jan. 2012.

[2] Y. Kyong and T. G. Robertazzi, "Greedy Signature Processing with Arbitrary Location Distributions: A Divisible Load Framework," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, no. 4, pp. 3027-3041, October 2012.

[3] S. Suresh, C. Run, H. J. Kim, T. G. Robertazzi and Y. I. Kim, "Scheduling Second-Order Computational Load in Master-Slave Paradigm," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, no. 1, pp. 780-793, Jan. 2012.

[4] M. Hu and B. Veeravalli, "Dynamic Scheduling of Hybrid Real-Time Tasks on Clusters," *IEEE Transactions on Computers*, 19 Aug. 2013.

[5] L. Marchal, Y. Yang, H. Casanova, and Y. Robert, "A Realistic Network/Application Model for Scheduling Divisible Loads on Large-Scale Platforms," *Proceedings. of the 19th IEEE International Parallel and Distributed Processing Symposium*, 04-08 April 2005.

[6] T. G. Robertazzi, "Processor equivalence for daisy chain load sharing processors," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 29, no. 4, pp. 1216-1221, Oct. 1993.

[7] O. Beaumont, H. Casanova, A. Legrand, Y. Robert, and Y. Yang, "Scheduling divisible load on star and tree networks: results and open problems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 3, pp. 207-218, March 2005.

[8] V. Bharadwaj, H. F. Li and T. Radhakrishnan, "Scheduling divisible load in bus networks with arbitrary processor release times", *Computers and Mathematics with Applications*, vol. 32, no. 7, pp. 57-77, October 1996.

[9] K. Li, "Scheduling divisible tasks on heterogeneous linear arrays with applications to layered networks," *Proceedings of International Parallel and Distributed Processing Symposium. (IPDPS 2002)*, 15-19 April 2001.

[10] A. Ghatpande, H. Nakazato, H. Watanabe and O. Beaumont, "Divisible Load Scheduling with Result Collection on Heterogeneous Systems," *2008 IEEE International Symposium on Parallel and Distributed Processing. (IPDPS 2008)*, pp.1-8, 14-18 April 2008.

[11] J. Blazewicz and M. Drozdowski, "Scheduling divisible jobs on hypercubes," *Parallel Computing*, vol. 21, no. 12, pp. 1945-1956, Dec. 1995.

[12] J. Blazewicz and M. Drozdowski, "The Performance Limits Of Two-Dimensional Network of load-sharing processors," *Foundations of Computing and Decision Sciences*, vol. 21, pp. 3-15, 1996.

[13] K. Li, "Speed-up of Parallel Processing of Divisible Loads on k-dimensional Meshes and Tori," *The Computer Journal*, vol. 46, no. 6, pp. 625-631, Jan. 2003.

[14] K. Li, "Improved Methods for Divisible Load Distribution on k-Dimensional Meshes Using Pipelined Communications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, no. 12, pp. 1250-1261, December 2003.

[15] M. Drozdowski and W. Glazek, "Scheduling divisible load in a three-dimensional mesh of processors," *Parallel Computing*, vol. 25, no. 4, pp. 381-404, April 1999.

[16] J. Blazewicz, M. Drozdowski, F. Guinand and D. Trystram, "Scheduling a divisible task in a two-dimensional toroidal mesh," *Discrete Applied Mathematics*, vol. 94, no. 1C3, pp. 35-50, 15 May 1999.

[17] B. Veeravalli, X. Li and C. Ko, "On the Influence of Start-Up Costs in Scheduling Divisible Loads on Bus Networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 11, no. 12, pp. 1288-1305, December 2000.

[18] J. Hu and R. Klefstad, "Scheduling Divisible Loads on Bus Networks with Arbitrary Processor Release Time and Start-Up Costs: XRMI," *2007 IEEE International Performance, Computing, and Communications Conference (IPCCC 2007)*, pp. 356-364, 11-13 April 2007.

[19] S. Suresh, V. Mani and S. N. Omkar, "The effect of start-up delays in scheduling divisible load on bus networks: An alternate approach," *Computers and Mathematics with Applications*, vol. 46, no. 10C11, pp. 1545-1557, NovemberCDecember 2003.

[20] Y. Chang, J. Wu, C. Chen and C. Chu, "Improved Methods for Divisible Load Distribution on k-Dimensional Meshes Using Multi-Installment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 11, pp. 1618-1629, Nov. 2007.

[21] M. Shang and S. Sun, "Optimal multi-installments algorithm for divisible load scheduling," *2005. Proceedings. Eighth International Conference on High-Performance Computing in Asia-Pacific Region*, July 2005.

[22] Y. Yang, K. van der Raadt and H. Casanova, "Multiround Algorithms for Scheduling Divisible Loads," *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 11, pp. 1092-1102, Nov. 2005.

[23] M. A. Moges, D. Yu and T. G. Robertazzi, "Grid scheduling divisible load from two sources," *Comput. Math. Appl.* vol. 58, no. 6, pp., 1081-1092, Sept. 2009.

[24] J. Jia, B. Veeravalli and J. Weissman, "Scheduling Multisource Divisible Loads on Arbitrary Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 4, pp. 520-531, April, 2010.

[25] T. G. Robertazzi and D. Yu, "Multi-Source Grid Scheduling for Divisible Loads," *2006 40th Annual Conference on Information Sciences and Systems*, pp. 188-191, 22-24 March 2006.

[26] C. Martinez, R. Beivide, E. Stafford, M. Moreto and E. M. Gabidulin, "Modeling toroidal networks with the Gaussian integers," *IEEE Transactions on Computers*, vol. 21, no. 8, pp. 1132-1142, Aug. 2010.

[27] M. Flahive and B. Bose. "The topology of Gaussian and Eisenstein-Jacobi interconnection networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 21, no. 8, pp. 1132-1142, Aug. 2010.

[28] Z. Zhang, Z. Guo and Y. Yang, "Efficient All-to-All Broadcast in Gaussian On-Chip-Networks," *IEEE Transactions on Computers*, vol. 62, no. 10, pp. 1959-1971, Oct. 2013.

[29] Z. Zhang, Z. Guo and Y. Yang, "Bufferless Routing in Optical Gaussian Macrochip Interconnect," *IEEE Transactions on Computers*, 02 July 2013.

[30] Z. Zhang, Z. Guo and Y. Yang, "Bufferless Routing in Optical Gaussian Macrochip Interconnect," *2012 IEEE 20th Annual Symposium on High-Performance Interconnects (HOTI)*, pp. 56-63, 22-24 Aug. 2012.

[31] G. H. Hardy and E. M. Wright, *An introduction to the theory of numbers*, First edition, Clarendon Press, Oxford, 1938.

[32] A. Shamaei, B. Bose and M. Flahive, "Higher dimensional Gaussian Networks," *Proceeding of IEEE International Parallel and Distributed Processing Symposium Workshops*, pp. 1438-1447, May 2014.

[33] N. Megiddo, "On the complexity of linear programming," *Advances in economic theory*, vol. 12, pp. 225268, Cambridge Univ. Press, Cambridge, 1989.

[34] http://en.wikipedia.org/wiki/Ikonos

[35] http://www.design-reuse.com/articles/10496/a-comparison-of-network-on-chip-and-busses.html

AUTHOR BIOGRAPHIES

Zhemin Zhang received the B.Eng. in electrical engineering from Huazhong University of Science and Technology, Wuhan, China, in 2009. He has been working towards the Ph.D. degree in the Department of Electrical and Computer Engineering, Stony Brook University, New York since 2009. His research intrests include parallel and distributed processing, data center network, high speed optical network, on-chip network, etc.

Thomas G. Robertazzi received the Ph.D from Princeton University, Princeton, NJ, in 1981 and the B.E.E. from the Cooper Union, New York, NY in 1977. He is presently a Professor in the Dept. of Electrical and Computer Engineering at Stony Brook University, Stony Brook N.Y. He has published extensively in the areas of parallel processing scheduling, telecommunications and performance evaluation. Prof. Robertazzi has also authored, co-authored or edited six books in the areas of networking, performance evaluation, scheduling and network planning. He is a Fellow of the IEEE and since 2008 co-chair of the Stony Brook University Senate Research Committee.